NBS-GCR-87-538

SEES: An Expert System for the Strength Evaluation of Existing Structural Members

Joseph Francis Peters

October 1987

Issued January 1988

Prepared for
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Building Technology
Gaithersburg, MD 20899

NBS-GCR-87-538

SEES: AN EXPERT SYSTEM FOR THE STRENGTH EVALUATION OF EXISTING STRUCTURAL MEMBERS

Joseph Francis Peters
Master of Science in Civil Engineering Candidate

Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

October 1987

NBS Cooperative Agreement 70NANB5H0584

Issued January 1988

Prepared for
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Center for Building Technology
Gaithersburg, MD 20899

CARNEGIE MELLON UNIVERSITY

SEES:

An Expert System For

the

Strength Evaluation of Existing Structural Members

Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Civil Engineering

By

Joseph Francis Peters
B.S.C.E. - Michigan Technological University

Department of Civil Engineering
Carnegie-Mellon University
Pittsburgh, Pennsylvania

October, 1987

Abstract

This thesis is a report of the design and implementation of the SEES expert system; a knowledge based system for strength evaluation of existing structural members. The expert system provides engineering knowledge for in situ member evaluation compiled into an interactive computer program to aid in the solution of a characteristically uncertain engineering problem. Not intended to replace engineers, the system's purpose is twofold: provide engineering judgment in a computer program, and attend to details that engineers may let go unnoticed.

Two studies made up the work of this thesis: the use of an artificial intelligence programming environment for an engineering application and the study of expert domain knowledge needed for strength evaluation. The resulting computer program is a goal driven expert system developed in a PC-AT version of OPS5, where the problem solution involves the satisfaction of three top level goals: formulation, redesign, and evaluation. Formulation provides the problem definition. Redesign requests or infers design parameters. Evaluation uses current codes to assess the capacity of the existing member. The study of expert domain knowledge involved learning the procedures and rules of engineering judgment used in the evaluation of flexural, shear, and torsional capacity of reinforced concrete T, spandrel, and rectangular beam sections.

The result of the study is a generic engineering expert system control structure readily expandable to at least seven other strength evaluation problems and a prototype implementation for reinforced concrete beams.

Acknowledgments

I would like to thank the following individuals at Carnegie Mellon University for their help and counseling on this project. Professor Mary Lou Maher for her advice during the entire project, and her courses on Computer Methods in Civil Engineering and Computer Aided Engineering. Professor Steven J. Fenves for his help in defining the engineering problem, his input of expert domain knowledge, and his course on Expert Systems in Civil Engineering.

I would like to thank the following individuals at the National Bureau of Standards. Dr. Kent A. Reed for his hospitality during my stay at the bureau, his valuable advice regarding both the computer program and my approach to the project as a whole, his additional insight into many interesting issues regarding the implementation and his teaching of related computer aided engineering ideas during the course of the summer. Dr. Charles Scribner for providing the majority of the domain knowledge for the system regarding both problem solution knowledge and reinforced concrete design knowledge. William F. Danner for his help translating important OPS5 functions from Franzlisp into Golden Common Lisp and his tutoring on many ideas in artificial intelligence.

I would also like to thank the following. Dr. Chester P. Seiss at the University of Illinois for his help in my search for past editions of the ACI code. Friends at both Carnegie Mellon and the National Bureau of Standards for putting up with all of my questions.

I would like to thank Michelle, for her confidence in me and her patient waiting the last 13 months.

Table of Contents

List of Figures

List of Tables

Chapter One
Introduction

This document is a report on the design and implementation of the SEES (Strength Evaluation of Existing Structures) expert system. The first chapter is an introduction to the rest of the document, providing the document scope, an annotated contents, and the conventions used throughout the document.

1.1. Scope of Document

The objective of this document is to report on the development of an expert system for the evaluation of strength of existing structural members. The report focuses on two major studies needed to develop the expert system. One study is the implementation of the engineering problem in a production rule environment and the second is a study of the expertise needed to solve the strength evaluation of existing structures. It is hoped that after reading this report, another structural engineer could pick up where this project left off, and further develop the SEES expert system into a practical tool.

1.2. Annotated Contents

After this introductory chapter, the rest of the document is composed of the following chapters. Chapter 2 is devoted to a detailed explanation of the problem and its role in structural engineering. The discussion includes the problem definition, its importance in structural engineering, and the methods and sources of expertise needed to solve the problem of in situ strength evaluation. Chapter 3 gives an overview of the SEES expert system, starting with an explanation of the motivation and justification for the use of an expert system to solve the problem, describing the scope of the system in terms of the input and output, and providing the problem solution paradigm and the individual sources of domain knowledge that were used to obtain the solution. Chapter 4 summarizes the programming environment by describing the OPS5 programming language, the GC-Lisp software in which OPS5 runs, and the IBM PC-AT hardware on which the environment exists. Chapter 5 gives the system design, focusing on the design of the process knowledge, domain knowledge, and working memory. Chapter 6 cites actual expert knowledge and heuristics that are incorporated into the system. Chapter 7 provides

a summary and describes the assumptions and limitations to the existing prototype of the system, along with possible enhancements and an evaluation of the prototype. Chapter 8 gives conclusions. References, example runs and portions of the code are in the Appendices.

1.3. Conventions Used

The following section defines the conventions used in this document, as special definitions, notations, acronyms, and terminology that will help the reader better understand the document. Those definitions not appearing in this section will be given when they occur in the document.

1.3.1. Definitions

Control strategy is that part of an expert system provided by the programming language which governs the process the system uses in getting from its initiation to final state.

Design parameter is a piece of data needed to describe the design of a structural member. It could be a numerical value used in computation or a qualitative value that helps in making a decision about other parameters.

Domain knowledge is knowledge that is incorporated into the knowledge base of an expert system that reflects the knowledge of the specialty area from which the problem was taken.

Process knowledge is knowledge in the form of rules that manages the system control strategy.

Problem type is one of eight possible structural engineering design problems; those being beams and columns made of either steel, reinforced concrete, composite construction, or timber.

Problem definition is specification of the strength parameter to be checked and any special criteria on any one of the eight problem types defined above.

1.3.2. Acronyms

SEES - Strength Evaluation of Existing Structures; name of the expert system, taken from the corresponding chapter of the American Concrete Institute Building Regulations for Reinforced Concrete.

ACI - American Concrete Institute

AISC - American Institute of Steel Construction

GCL - Golden Common Lisp; the lisp environment in which OPS5 runs.

LHS - Left Hand Side; the part of a production rule which contains the conditions.

RHS - Right Hand Side; the part of a production rule which contains the actions.

rc - reinforced concrete

wme - working memory element; the term used for an instance of an OPS5 working memory class.

1.3.3. Notations

Notation is as follows:

M	- the designator for the moment capacity of a structural member.
V	- the designator for the shear capacity of a structural member.
T	- the designator for the torsional capacity of a structural member.
P	- the designator for the axial capacity of a structural member.

Other notation will be defined as needed.

1.3.4. Terminology

system	throughout the rest of the document, "...the system..." refers to the SEES expert system. "The system" and "SEES" will be used synonymously.
OPS5	the name of the expert system programming language in which SEES is implemented.
rebar	short name used for reinforcing bar.
rule,production	these two terms will be used interchangeably. They both pertain to rules in the OPS5 language.

Chapter Two
Strength Evaluation

This chapter describes the strength evaluation problem, explaining why there is a need to solve such a problem and how the problem is solved by experts.

2.1. Problem Definition

Most buildings are designed for a forty year life span. In reality, some buildings last for much longer periods while some are destroyed and replaced with new buildings. Whatever the case may be, a building's life is dynamic and it is not uncommon to have a particular component or set of components re-analyzed by a consulting engineer during the erected life of a building. Questions regarding the capacity of individual members of buildings arise in new buildings as well as old. Old buildings may require analysis to prove adequacy for continued use or determine strengthening requirements for stricter regulations such as new occupancy or earthquake criteria. New buildings may need analysis for feasibility studies of structural member "add-ons". Engineering of structural members does not stop after a building has been built.

The concern of building owners and structural engineers, during design and redesign of buildings, is satisfying relevant codes and specifications, thereby assuring the safety of the occupants. Therefore when a question is raised about the capacity of a structural member, that member must be analyzed to ensure it will continue to maintain the structural integrity of the entire building. Existing structural members are analyzed to determine whether a member should see continued service, and if so, determine whether a member is adequate as is or needs strengthening.

This study addresses the assessment of individual structural members for strength criteria, like flexure, shear, torsion, axial, and axial-bending with respect to present code specifications. The procedure for assessing existing structural members is much like design of new members except the member is in situ and new design parameters need not be determined but existing ones inferred and checked for adequacy. With the strength evaluation of existing structural members, decisions can be made about the integrity of the member and continued use of the structure.

An example of a member considered for an alternate use arose recently in Pittsburgh. The

owner of an old building was offering space in the building for rent. The building was built in the 1920's and was originally used as a garage. An environmental researcher became interested in the space for radon experiments, but stated that he would have to set up configurations of lead brick shielding weighing 12000 pounds. The location of the large brick loads would be centered over a 16" by 26" reinforced concrete beam as shown in Figure 2-1. Before the building owner could allow the researcher to move in, he would have to consult an experienced engineer who could make a judgment about the strength of the member for both flexure and shear. To make such a judgment, would require experience with evaluation of existing structures. Also, knowledge of design practice in the 1920's would help. This example will be referred to again later.

2.2. Method of Solution

The philosophy behind the evaluation of existing structures for strength criteria is summarized in Section 1 of Chapter 20 of the ACI code: "If doubt develops concerning the safety of a structure or member, the Building Official may order a structural strength investigation by analysis or by means of load tests, or by a combination of analyses and load tests." [ACI 19 83] Although the code leaves open the possibility of both analytical and load test analysis, it is evidenced by the statements in Section 20.5 of the ACI code [ACI 19 83] that load testing be limited to flexural members. Only analytical techniques are addressed in this project.

In order to evaluate the strength of an existing member, an engineer must first determine the value of the relevant design parameters and second, evaluate the strength criteria specified in current codes. The more formidable of the two tasks is the acquisition of design parameter values. Section 20.2 of ACI 318-83, Analytical investigations - General, states: "If strength evaluation is by analysis, a thorough field investigation shall be made of dimensions and details of members, properties of materials, and other pertinent conditions of the structure as actually built." [ACI 19 83] When faced with this acquisition problem, an engineer follows a sequence of steps. Design drawings are sought out first so that the actual design information may be obtained. For some buildings, design drawings are available, while for others, typically old buildings, design drawings are not readily attainable. In the event that design drawings are not available, the engineer must rely on experience and documented information on design practice when the building was designed to estimate design parameter values. In addition to drawings and documents, the engineer inspects the component visually to aid in determining design parameters. In the case of reinforced concrete members, this is especially difficult since it is impossible to see into a concrete member to determine reinforcing details. Even in steel structures it is difficult to tell by looking what material strength steel was used. There are many cases in which engineers must settle for an estimate of design parameter values.

The objective of the second part of the evaluation problem is to analyze the structural member

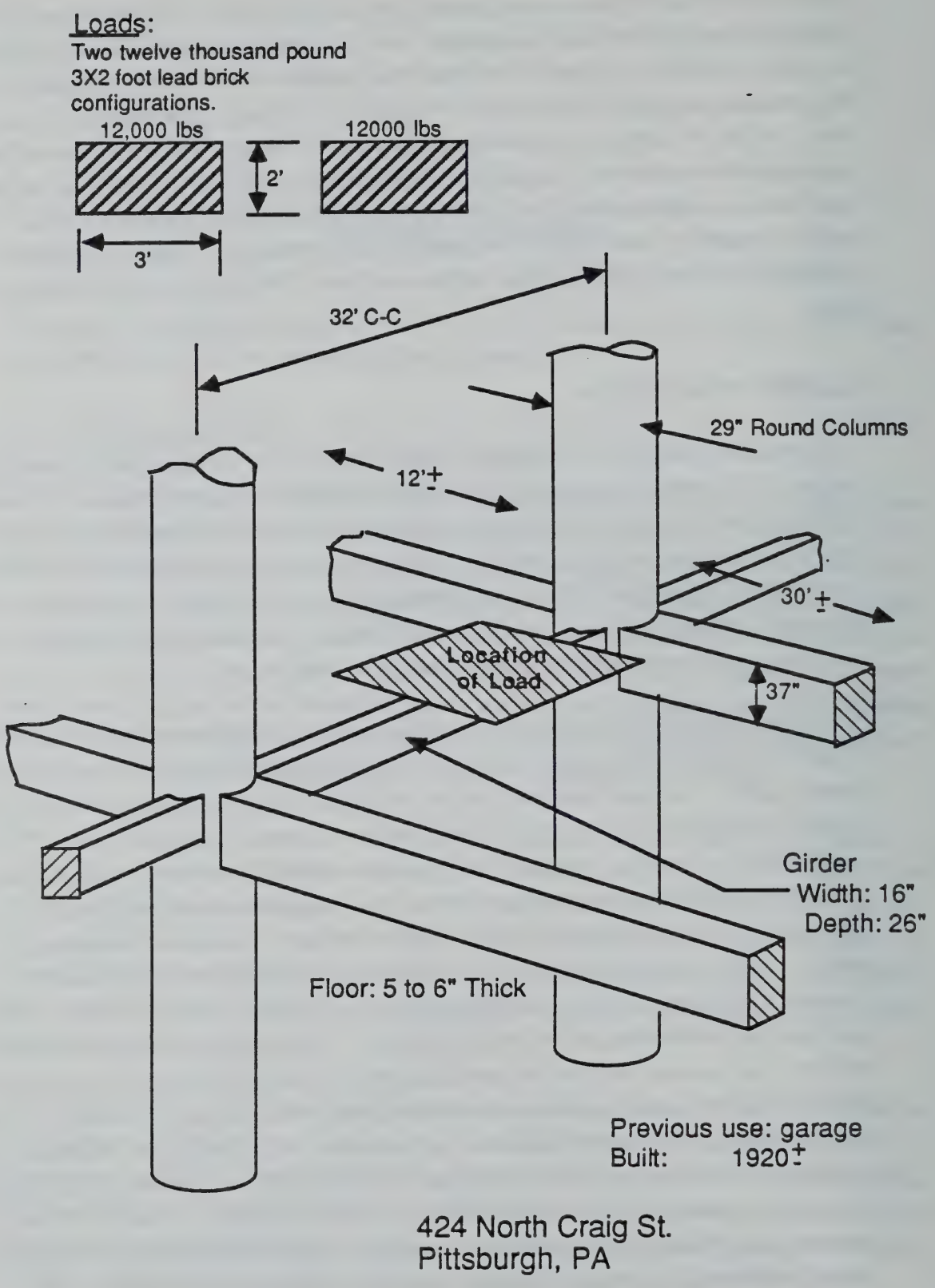

Figure 2-1: Example of Existing Member In Need of Evaluation

using the criteria of the current editions of the codes, the ACI for concrete and the AISC for steel. This portion of the problem involves using the parameter values obtained to compute values of criteria test ingredients, like moment capacity, and comparing allowable code provisions against actual conditions.

Chapter Three
Overview of SEES

This chapter provides an overview of the SEES system by presenting the motivation and justification for using an expert system approach to solve the SEES problem, the scope of the system in terms of input and output, comments on the uncertainty of the system output, the system's basic behavior, and the problem solving method and decomposition.

3.1. Motivation and Justification

3.1.1. Motivation

The motivation for development of the SEES expert system is to have an interactive system that aids engineers faced with the structural evaluation problem. SEES should be used in conjunction with the experience that the engineer already has in such problems, and when necessary provide the expertise needed where the engineer's own experience may be lacking. The system should also be used as an aid to engineers by giving attention to details that might otherwise go unattended in an analysis. The motivation for the system is not to replace engineers, but to enhance his ability to evaluate the strength of existing structural components.

3.1.2. Peripheral Motivation

A peripheral motivation for SEES is related to building design database projects, that is, projects constructing databases of pertinent information about a building for the entire life of the building. The portion of SEES that determines design parameters can be used to generate building databases for structures that already exist. A good example of this use is for as-built databases for old buildings. This would be of use for buildings in the eastern half of the United States that were never designed for strict seismic loading conditions, but may eventually need evaluation for such conditions. Due to the belief that portions of the eastern United States may become seismically active in the next twenty five years, being able to evaluate existing structures for earthquake criteria and generating databases for buildings for strengthening requirements, would be of crucial importance.

3.1.3. Justification

To justify the use of expert system techniques for the SEES implementation, a statement from the book Building Expert Systems is relevant: "...when the knowledge is subjective, ill-codified, and partly judgmental, expert systems embodying a heuristic approach are more appropriate". [Buchanan et al 83] An heuristic approach is distinguished from an algorithmic one. Justification for expert system techniques is made by addressing each of the three characteristics mentioned in the above citation. SEES is subjective because conclusions are drawn from the knowledge base about indefinite problem information depending on the conditions of the problem supplied by the user and the knowledge base, reflecting the experience of a structural engineer. SEES is ill-codified because the engineering problem does not have a formalized solution; only guidelines for solution. The judgmental characteristic of SEES is due to the expertise and experience that it contains, such as knowledge of the history of changes to code specifications, knowledge of how to infer the value of unknown design parameters, and knowledge of what test criteria are important to a defined problem.

Further justification for the use of expert system techniques over procedural programming for the SEES implementation pertains to the accuracy of strength evaluation in general. Professional engineers are aware that codes like the ACI and AISC normally expect inaccuracies up to ten percent when determining structural strength. Due to this expectation, engineers feel there is little use in the exact numerical computation of structural strength using six digit accuracy provided by a standard numerical program. An engineer should be able to apply engineering judgment when he feels that exact computation is not necessary. These engineer's philosophy is that "if you are not going to be right, you might as well not be right the easy way." In this lies the justification for an expert system approach to SEES: it allows for easy application of such engineering judgment when exact computation is not required or appropriate.

3.2. Scope

This section discusses the space within the problem domain that the system works, the input and output of the system, and the uncertainty of the system output.

3.2.1. Extent In Domain

A complete SEES expert system is intended to evaluate beam and column structural components that are made of either steel, reinforced concrete, composite construction, or timber, considering one structural member in a given execution. Within the evaluation of a single member, SEES evaluates one of the set of capacity parameters associated with the member, those being flexure, axial, shear, torsion, and combined axial and flexure. For example, SEES can evaluate the shear capacity of a reinforced concrete beam in one run, but must be run again to evaluate flexural capacity of the same member.

The methods of evaluation that SEES uses are the working stress method for steel members and timber members, and the ultimate strength method for concrete members. These methods are used for all members regardless of their date of construction. Essentially, all members are checked according to the philosophies of current design codes regardless of their original design philosophies.

3.2.2. Input

Input to the system is in the form of progressive user query of single pieces of data until the system can perform actions on the problem definition. Input takes one of two forms; problem specification input that defines the type of component and capacity and specific problem data that provides information for inferring design parameters.

3.2.2.1. Problem Specification Input

The problem specification input includes type of member to be evaluated, the material of the member, the capacity parameter the user wishes to know, and various other problem specification information needed regardless of problem type. Because of the nature of this input, the system assumes the user is able to provide all data items and is not prepared to handle cases in which the user does not know a certain piece of data. The problem specification input is summarized in Table 3-1 below.

INPUT ITEM	CHOICES
member type	beam or column
material	steel, reinforced_concrete, composite, or timber
capacity desired	flexure, shear, torsion, axial, or axial-bending
location of capacity	at_ends or midspan
span type	simple, fixed, hybrid, or cantilever
exposure	indoor or outdoor
loading	seismic or in_service

Table 3-1: Problem Specification Input

Some constraint checking is done in order to insure validity of user input. For example, it is

impractical to check the axial capacity of a beam. When the user inputs inconsistent data, the system usually allows the user to re-input new data.

3.2.2.2. Specific Problem Input

Specific problem input includes the relevant design parameters known by the user. Since there are two member types, four material types, and five possible capacities to be computed, there are many different problem definitions in which data items must be obtained. Once the problem specification input has provided the problem definition, the system knows what information it needs to evaluate the strength of the component.

The philosophy behind querying the user for design parameters is to insure use of known data before inferring the data. Of course there are parameters only the user can supply, like physical member dimensions and year of construction. There are also parameters for which it is impractical to query the user that are usually computed directly. Categories of specific problem data are summarized in Table 3-2 below.

Specific Problem Data Item	Examples
member dimensions	height, width, length
material properties	yield strengths and moduli of elasticity
quantities	areas and quantities of member components
special criteria	about physical condition of member

Table 3-2: Summary of Problem Specific Input

Four categories make up the specific problem input. Member dimensions supply spatial data about the physical member. Material properties supply mechanical data about the member's constituent materials. Quantities provide numbers of member components, like number of reinforcing bars in reinforced concrete design. Special criteria provide information about the physical condition of the member in terms of aging and damage.

Allowing the user to input as much about a problem as possible gives a range to the characteristic of the results generated by SEES. At one extreme the user is able to supply all information about the problem and SEES only calculates the strength of the member. At the other extreme the user can supply only the minimum required input and has to rely on the

expertise of SEES to infer the remaining parameters. Other cases lie in between the two extremes where responsibility for parameter acquisition is held by both parties.

3.2.3. Output

System output comes in the form of execution messages on system status, intermediate results, and final test results. The execution messages and intermediate results the system provides usually report an action the system is about to take or is taking, describe parameters to be acquired, or echo results of a particular parameter acquisition. Some examples are shown below.

- Parameter names are written out that are relevant to the problem definition.
- The system always states when it is inferring a parameter.
- The system always states when it will compute a parameter directly.
- The user is warned that the system will halt if the user does not know parameter that cannot be inferred.
- The system always lets the user know if it is about to halt.
- The user will be asked if he wishes to continue after specifying that he does not know a parameter that cannot be inferred.
- The system writes the value it inferred for a parameter and asks if the user wishes to override the value.
- The system writes the value of a parameter it computed from other lower level parameters.
- The system writes out the names of test criteria that are required in order to evaluate the specified capacity.
- The system writes the value of an ingredient obtained for a criterion test.

The system also provides the final results of the criteria tests for the problem as satisfied or not satisfied. For example, the criteria that need to be checked for flexure of a reinforced concrete beam are:

- that the reinforcement ratio is between the minimum allowable and seventy-five percent of the balanced reinforcement ratio,
- that a percent of the nominal moment capacity of the beam is greater than the factored applied moment, and
- that the deflection criteria are met for the beam.

The system reports to the user that these are the criteria and checks each, reporting the results of each test, and then provides a final evaluation of the member.

3.2.4. Uncertainty of Output

A comment should be made about the uncertainty of the results obtained by SEES. It is important to know at which point the results generated by SEES should be reconsidered. Essentially, two results can be generated by SEES: one that the capacity of the member is acceptable with respect to present code criteria and the other that the capacity is unacceptable. The uncertainty depends on the amount of data the user can supply and how much is inferred. The results of SEES are categorized in Figure 3.1, where the range of user knowledge is compared to the results of SEES.

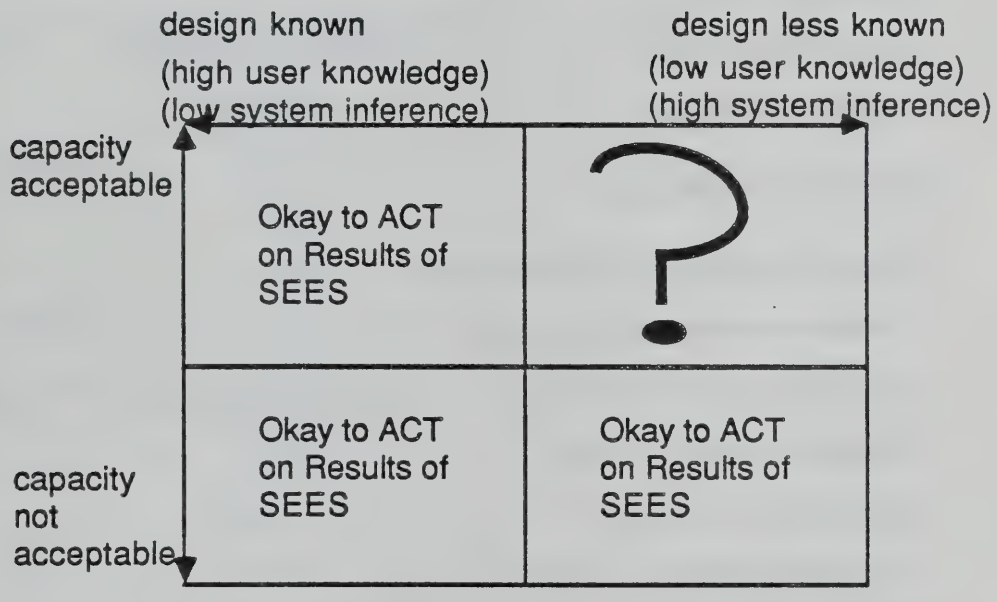

Figure 3-1: SEES Reliability as per discussion with Dr. K. Reed, NBS

Uncertainty of the results increases when the system infers the value of most parameters, and decides that the structural member is adequate. SEES essentially takes a problem full of unknowns and tells the user that the member is safe. The question mark in the figure indicates this case, and emphasizes that in this situation the results of SEES must be externally verified.

To partly overcome this problem is to design SEES as conservative as possible when determining design parameter values, in which case the system will tend to make more judgments on the "capacity unacceptable" side of results. The philosophy here is that it is better to report something is wrong when it is not, than to ignore a potentially dangerous situation.

3.2.5. Basic Behavior On User Level

This section provides a sample run of SEES to show the system's basic behavior. The problem is to evaluate the torsional capacity of a T shape reinforced concrete beam.

The problem is specified by the following information.

Problem Specification Information:

- member type = beam
- material type = reinforced concrete
- capacity to be checked = torsion at midspan
- span type = fixed
- environment = indoors
- loading = in service

Specific Problem Information:

- method of construction = monolithic pour
- location of member in slab = interior
- year built = approximately 1965
- beam web width = 10 inches
- span length = 360 inches
- bay spacing of beams = 300 inches
- total height of beam = 30 inches
- load = point load of 12000 pounds
- location of load = 9 inches off beam centerline

Photo Log of System Run

```
**GREETINGS**
THIS IS THE SECOND PROTOTYPE OF -SEES-

-- A KNOWLEDGE BASED EXPERT SYSTEM NAMED AFTER CHAPTER 20 OF THE ACI CODE
THAT HELPS IN THE EVALUATION OF STRENGTH OF IN SITU STRUCTURAL MEMBERS

THIS CURRENT PROTOTYPES CAPABILITY IS - REINFORCED CONCRETE BEAMS-

IN FORMULATE_PROBLEM GOAL


QUERY FOR BASIC MEMBER INFORMATION
Please submit the following information needed for the object: MEMBER_INFO
What type of member do you wish to evaluate?
   ANSWERS     CORRESPONDING VALUES

      1 ..... BEAM

      2 ..... COLUMN

Please input the number corresponding to your choice: 1

What material is the member composed of?
   ANSWERS     CORRESPONDING VALUES

      1 ..... STEEL

      2 ..... RC_CONCRETE

      3 ..... COMPOSITE

      4 ..... TIMBER

Please input the number corresponding to your choice: 2

What type of span is the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... SIMPLE

      2 ..... FIXED

      3 ..... HYBRID

      4 ..... CANTILEVER

Please input the number corresponding to your choice: 2


IN FORMULATE_PROBLEM GOAL

QUERY FOR PROBLEM DEFINITION
Please submit the following information needed for the object: PROBLEM_DEF
What capacity element are you interested in checking?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AXIAL

      2 ..... FLEXURE

      3 ..... SHEAR
```

```
      4 ..... TORSION

      5 ..... BEAM_COLUMN

Please input the number corresponding to your choice: 4

Where on the member do you need to know this?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AT_ENDS

      2 ..... MIDSPAN

Please input the number corresponding to your choice: 2


IN FORMULATE_PROBLEM GOAL

QUERY FOR IMPORTANT CRITERIA
Please submit the following information needed for the object: PROBLEM_CRITERIA
What type of loading do you wish to consider?
   ANSWERS     CORRESPONDING VALUES

      1 ..... IN_SERVICE

      2 ..... SEISMIC

Please input the number corresponding to your choice: 1

What is the exposure of the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... INDOOR

      2 ..... OUTDOOR

Please input the number corresponding to your choice: 1


THE FORMULATE PROBLEM GOAL IS NOW COMPLETE
IN REDESIGN GOAL

THE SHAPE OF THE MEMBER MUST BE KNOWN IN ORDER TO IDENTIFY RELEVANT PARAMETERS
THE POUR INFORMATION MUST BE KNOWN TO DETERMINE THE SHAPE OF THE CONCRETE BEAM
Please submit the following information needed for the object: POUR_INFO
How is the slab connected to the beam?
   ANSWERS     CORRESPONDING VALUES

      1 ..... MONOLITHIC

      2 ..... DETACHED

      3 ..... NO_SLAB

      4 ..... SHEAR_CONNECTORS

Please input the number corresponding to your choice: 1


LOCATION OF MEMBER UNDER SLAB IS IMPORTANT
Please submit the following information needed for the object: LOCATION_INFO
```

```
What is the location of the member within the structure?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  INTERIOR

      2 ..............  EDGE

Please input the number corresponding to your choice: 1


MEMBER SHAPE WILL BE TREATED AS A T SECTION

THE RELEVANT PARAMETERS FOR FLEXURE OF THIS FLANGED BEAM ARE
BW - THE WIDTH OF THE MEMBER WEB IN INCHES
HF - THE SLAB THICKNESS OVER THE BEAM IN INCHES
L - THE LENGTH OF THE MEMBER IN INCHES
BAY_SPACING - THE CLEAR DISTANCE TO THE NEXT BEAM IN INCHES
H - THE HEIGHT OF THE MEMBER IN INCHES
OVERHANG - THE OVERHANG OF THE EFFECTIVE FLANGE IN INCHES
SIG_X_SQUARED_Y - THE TORSIONAL SECTION PROPERTY OF THE BEAM IN CUBIC INCHES
F_PRIME_C - THE CONCRETE YIELD STRENGTH IN PSI

SEE IF USER KNOWS THE VALUE OF BW
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: BW
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: BW
Please input the value of this parameter
 making sure the units comply with pounds and inches please
10.0



SEE IF USER KNOWS THE VALUE OF HF
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: HF
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2
```

```
IT IS IMPERITIVE THAT THE USER BE ABLE TO SUPPLY THE
VALUE OF HF WHICH IS NOT INFERABLE
*OTHERWISE IT CANNOT COMPLETE ITS OBJECTIVE
*YOU WILL HAVE TO SUPPLY THIS INFORMATION FOR THE SYSTEM TO RUN
Please submit the following information needed for the object: CONTINUANCE
Do you wish to continue this session with SEES?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


HF WILL BE RE-ASKED FOR
Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: HF
Please input the value of this parameter
 making sure the units comply with pounds and inches please
6


SEE IF USER KNOWS THE VALUE OF L
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: L
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: L
Please input the value of this parameter
 making sure the units comply with pounds and inches please
360


SEE IF USER KNOWS THE VALUE OF BAY_SPACING
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: BAY_SPACING
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO
```

```
Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: BAY_SPACING
Please input the value of this parameter
 making sure the units comply with pounds and inches please
300



SEE IF USER KNOWS THE VALUE OF H
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: H
Do you know what this design parameter equals
                                        ?
   ANSWERS      CORRESPONDING VALUES

      1 ............... YES

      2 ............... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: H
Please input the value of this parameter
 making sure the units comply with pounds and inches please
30.0



SEE IF USER KNOWS THE VALUE OF OVERHANG
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: OVERHANG
Do you know what this design parameter equals
                                        ?
   ANSWERS      CORRESPONDING VALUES

      1 ............... YES

      2 ............... NO

Please input the number corresponding to your choice: 2



OVERHANG WILL HAVE TO BE INFERRED

THE FLANGE OVERHANG AS PER THE ACI CODE FOR SPANDRELS IS THE SMALLER OF
ONE EIGTH THE SPAN LENGTH MINUS THE WEB WITH OVER TWO WHICH IS 40.0 INCHES
EIGHT TIMES THE SLAB THICKNESS WHICH COMPUTES TO 48 INCHES
OR HALF THE CLEAR SPAN TO THE NEXT BEAM WHICH COMPUTES TO 150.0 INCHES
THE RESULT IS 40.0 INCHES
THE PARAMETER OVERHANG HAS BEEN INFERRED
TO A VALUE OF 40.0
```

```
DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
n


THE PARAMETER NAMED - SIG_X_SQUARED_Y - WILL BE COMPUTED DIRECTLY
INSTEAD OF FOLLOWING NORMAL ROUTINE OF USER QUERY
**COMPUTATION WILL OCCUR AS SOON AS THE CONSTITUANT DATA ITEMS BECOME AVAILABLE

THE TORSIONAL SECTION PROPERTY FOR THE T BEAM COMPUTES TO
5880.0 CUBIC INCHES

SEE IF USER KNOWS THE VALUE OF F_PRIME_C
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: F_PRIME_C
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2


F_PRIME_C WILL HAVE TO BE INFERRED

THE YEAR OF CONSTRUCTION MUST BE KNOWN FOR THE
ESTIMATION OF F_PRIME_C

SEE IF THE USER KNOWS WHEN THE STRUCTURE WAS BUILT

Please submit the following information needed for the object: YEAR_QUERY
Do you know the year give or take 5 that
the building was built?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


ASK USER WHEN STRUCTURE WAS BUILT

Please submit the following information needed for the object: YEAR_BUILT
What year was the building built? (1904 to now)
1965


FOR F_PRIME_C THE SYSTEM WILL ASSUME A VALUE OF 3500 PSI
FOR A BULDING BUILT AROUND 1965
THE PARAMETER F_PRIME_C HAS BEEN INFERRED
TO A VALUE OF 3500

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
```

```
Y OR N
Y

Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: F_PRIME_C
Please input the value of this parameter
 making sure the units comply with pounds and inches please
4000



***THE EVALUATION TEST CRITERIA FOR THE TORSIONAL
CHECK ON THE T RC MEMBER PROBLEM FOLLOW***
THERE IS ONLY ONE TEST FOR TORSION IN THE PRESENT PROTOTYPE

THE TEST IS FOR WHETHER TORSION NEED EVEN BE OF CONCERN IN THE MEMBER

AS PER ACI 318-83 PARAGRAPH 11-6-1 TORSION NEED NOT BE CHECKED WHEN THE
APPLIED TORQUE IS LESS THAN THE MEMBERS BASE CONTRIBUTION

NORMALLY TORQUE NEED NOT BE CONSIDERED

CRITERIA INGREDIENT TU TO BE OBTAINED THRU GET_PARAMETER METHODS

SEE IF USER KNOWS THE VALUE OF TU
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: TU
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: TU
Please input the value of this parameter
 making sure the units comply with pounds and inches please
96000



CRITERIA INGREDIENT TU HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED PARAMETER
TU WITH A VALUE OF 96000

BASE TORSION RESISTANCE OF MEMBER COMPUTED AS PER ACI CODE WITH
- A STRENGTH REDUCTION FACTOR OF 0.85
- A CONCRETE YIELD STRENGTH OF 4000 PSI
- A COMPUTED MEMBER TORSION PROPERTY SIGMA_X_SQUARED_Y OF 5880.0 CUBIC INCHES

THE BASE RESISTANCE EQUATES TO 1.58051F+05 INCH-POUNDS

***** EVALUATION CRITERIA TEST *****
BASE_TORSION_SUFFICIENT EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- MEMBER_TORQUE_RESISTANCE
```

```
IS GREATER THAN OR EQUAL TO INGREDIENT TU YIELDS RESULT OF
-- SATISFIED

********** ECHO PROBLEM DEFINITION **********
THE FIXED RC_CONCRETE T BEAM UNDER
IN_SERVICE LOADING CONDITIONS IN AN INDOOR ENVIRONMENT
HAS BEEN INVESTIGATED FOR A QUESTION OF TORSION AT MIDSPAN


IN SOLUTION TO THE SPECIFIED PROBLEM THE FOLLOWING RESULTS WERE
OBTAINED REGARDING THE CRITERIA RELEVANT TO THE DESIRED CAPACITY
*SYSTEM UNITS ARE POUNDS AND INCHES

THE TEST CRITERIA NAMED - BASE_TORSION_SUFFICIENT - WHERE IT IS DESIRED
THAT THE VALUE OF MEMBER_TORQUE_RESISTANCE BE GREATER THAN THE VALUE OF TU

HAS YIELDED THE FOLLOWING
- MEMBER_TORQUE_RESISTANCE HAS A VALUE OF 1.58051F+05
- TU HAS A VALUE OF 96000
THE RESULT OF THIS TEST BEING -- SATISFIED


-- SEES -- WILL NOW TERMINATE

THIS CURRENT VERSION OF SEES IS THE SECOND PROTOTYPE
AND APOLOGIZES FOR THE LACK OF ANTICAPATED CAPABILITIES
OF THE FULL INTENDED SYSTEM
end -- no production true
 104 PRODUCTIONS (1263 // 2201 NODES)
 68 FIRINGS (265 RHS ACTIONS)
 0 MEAN WORKING MEMORY SIZE (125 MAXIMUM)
 0 MEAN CONFLICT SET SIZE (6 MAXIMUM)
 0 MEAN TOKEN MEMORY SIZE (203 MAXIMUM)
```

The results are summarized below:

- When the user was not able to specify the thickness of the slab, the system warned him that it would halt unless he input a number, so the user assumed a 6 inch slab.
- The value of the concrete yield strength, f'_c, was inferred as 3500 psi when the user was unable to specify it. When the user saw what the system inferred though, he changed the value to 4000 psi.
- The base torsional resistance of the member was checked against the actual applied torque. The base torsional resistance was computed as 158,051 inch-pounds and the applied torque was 96,000 inch-pounds, indicating that the test was satisfied and the capacity of the member for torsion is adequate.

3.3. Solution Method

The following section discusses the solution method as decomposition. The objective of the SEES expert system is to: consider a particular structural member, the strength parameter of interest, and loading criteria, and determine the necessary design data and evaluate the member using relevant criteria tests.

To simplify the problem solution, the problem is decomposed into subproblems. These subproblems, much like the subproblems encountered in a design process, are:

1. formulation
2. redesign
3. evaluation

Figure 3-2 illustrates the problem decomposition of SEES; each major subproblem is described below.

3.3.1. Problem Formulation

The problem is formulated through the identification of three specifications: the type of member, the strength capacity that will govern the design, and the special considerations important to the evaluation of the member. The member type is specified as beam or column, support conditions, and material. The strength capacity is specified as flexure, shear, axial, torsional, or combined axial-bending capacity and location of capacity check. The special considerations include the loading conditions and the environment in which the member exists. Once these three problem specifications are known, the problem has been formulated and the next subproblem can be executed.

3.3.2. Redesign

Structural engineering design problems involve determining values for design parameters. The redesign subproblem has a similar goal. Although the system is not actually redesigning the structural member, since new design parameter values need not be determined, the actions it takes resemble those usually taken by an engineer in the design of a structural member. The redesign is done by first identifying the relevant parameters, and second, determining values of those parameters.

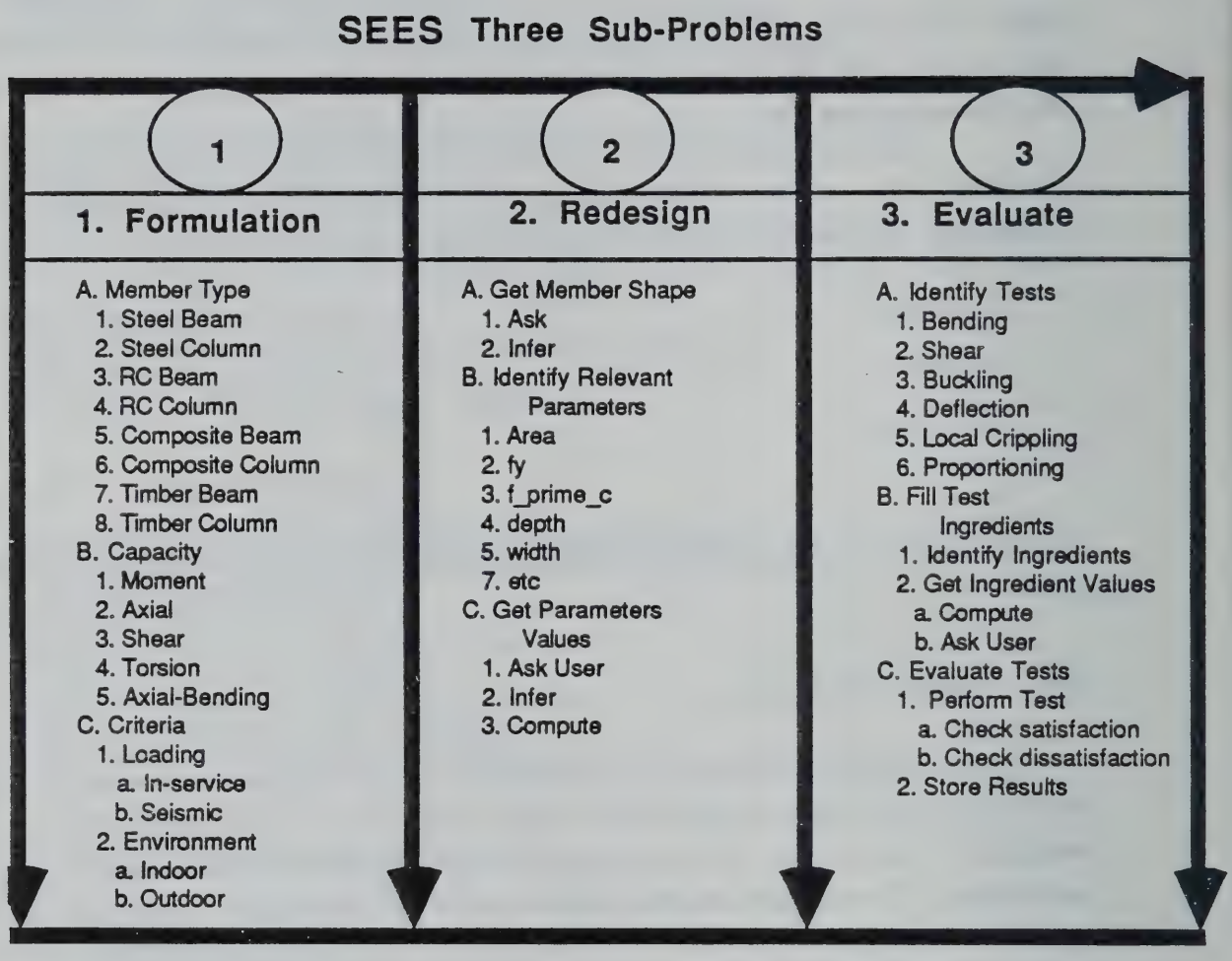

Figure 3-2: SEES Problem Breakdown

3.3.3. Evaluation

The evaluation subproblem identifies the relevant criteria for the evaluation of the desired strength parameter, determines the values of the criteria variables and performs the evaluation. For example, in order to evaluate the flexural capacity of a steel beam, extreme fiber stress, buckling of the flange, and lateral torsional buckling are checked to comply with the specifications of the AISC code. Values are assigned to the variables associated with these tests, like allowable steel stress, actual stress, web thickness, and flange thickness. Assigning values to criteria variables is similar to the redesign subproblem where the relevant variables are first identified and then values are sought out by inference or by user response. Finally, the system

determines the status of each criterion by comparing actual and allowable values. Upon completion of this testing, the system judges the member's capacity as a whole.

3.4. Sources of Domain Knowledge

The sources of expertise used for development of the system are Doctor Kent A. Reed and Doctor Charles Scribner at the National Bureau of Standards, Professor Chester P. Siess at the University of Illinois, and Professor Steven J. Fenves and Professor Mary Lou Maher at Carnegie Mellon University.

Literary sources of expertise are all past American Concrete Institute codes and Joint Committee Reports dating back to 1904, various design handbooks and texts from a comparable span of years, and any actual case histories available and pertinent to the knowledge base.

Chapter Four
Programming Environment

This chapter describes the programming environment in which SEES is implemented. The programming environment can be considered in three levels; the OPS5 expert system programming language in which SEES is written, the software on which OPS5 is written, and the machine on which the software environment resides.

4.1. Programming Language : OPS5

The expert system language in which SEES is implemented is the OPS5 programming language [Forgy 81]. OPS5 was originally written by Professor Charles Forgy at Carnegie Mellon University. OPS5 is classified as a production system language. The original version of OPS5 is written in Franzlisp. The version that is used for SEES is a translated version running in GC-Lisp.

This section discusses the components of the OPS5 expert system programming language, the control strategy provided by OPS5, and the peripheral functions written to enhance OPS5's user friendliness and capabilities.

4.1.1. Components of OPS5

OPS5 is a production based language where rules and facts are manipulated by the OPS5 *interpreter*. The rules comprise the *production memory* while the facts comprise the *working memory* of the OPS5 system. The three components of the OPS5 system are:

1. production memory
2. working memory
3. interpreter

Basically, the interpreter matches the left hand side *conditions* of the rules in production memory to facts in working memory creating a *conflict set* of rules that are all eligible to fire, chooses the best possible rule to fire from the conflict set, and executes the *actions* in the right hand side of that rule. The components of OPS5 will be discussed in the following paragraphs.

4.1.1.1. Production Memory

Production memory contains unordered production rules. Each rule contains a unique name along with a vector of *left hand side* (LHS) premises and a vector of *right hand side* (RHS) actions. An example of the syntax for writing OPS5 productions is as follows:

```
(p  a_rule              ;rule indicator "p" and rule name
    (premise one)       ;left hand side premise
    (premise two)       ;
    (.....      )       ;from one to n premises can be in
    (premise n)         ;one left hand side of a rule
    -->                 ;indicator that the right hand side
                        ;follows.
    (action one)        ; one to n actions
    (action two)
    (action ..)
    (action n)
)                       ;closing parenthesis to rule
```

4.1.1.2. Working Memory

There is only one data construct available in OPS5: working memory class. This construct uses object-attribute-value tuples. In OPS5, this data construct is a set of working memory *classes* that have an associated set of attribute-value pairs in a vector form. Many instances of these classes make up the working memory of the production system.

There are two uses of working memory elements in OPS5 production systems. One use is for declarative knowledge about the problem domain. This type of working memory is a portion of the permanent database of the production system. The second use is for dynamic knowledge about the current problem solution. Examples of the uses of working memory elements are given below.

A set of working memory elements can be loaded into working memory to comprise a portion of the permanent domain knowledge base of the expert system. For example, the history of changes made in the yield strength of steel through time might be represented as such:

```
(steel_history                  ;object class
        ^start_year 1904        ;A-V pair
        ^end_year 1934          ;A-V pair
        ^yield_strength 25000)  ;A-V pair
```

During execution, working memory elements are also used for creation and deletion of temporary

important data items needed for the problem solution [Forgy 81]. An example of a working memory element for factual information storage of a design parameter is shown below.

```
(parameter_information      ;object class

        ^name  beta_one     ;design parameter name

        ^inferable yes      ;parameter can be inferred

        ^criteria f_prime_c ;how it is inferred

        ^inferred nil       ;whether it has been inferred yet

        ^method i           ;if there is a special method of
                            ;obtaining this parameter

        ^over_ride nil      ;if user wishes to over ride an
                                ;inferred value

)
```

4.1.1.3. Interpreter

The recognize-act cycle for the OPS5 interpreter uses the following steps:

1. The interpreter finds the rules whose LHS premises match current working memory and inserts those rules into the conflict set.
2. It uses a conflict resolution strategy to choose one particular rule out of the conflict set.
3. Once the conflict resolution strategy has determined the dominating rule from the conflict set, it fires the rule by performing the actions defined in RHS of the rule.

OPS5 provides two conflict resolution strategies: LEX and MEA. The MEA will be discussed here since it is used for the SEES expert system. The dominating production in the conflict set is found by the following process of elimination.

1. Discard any productions that have already fired.
2. Compare the recency of the working memory elements in the first premise of each of the remaining productions in the set. The production using the most recent working memory elements is the dominating production.
3. If the recency criteria does not work for the first premises of the productions in the set, MEA considers the recency of the remaining premises until it finds recency.
4. If a single production does not dominate yet, the specificity rule is used, choosing the production having the largest number of premises.
5. Finally, the strategy involves a random pick of the rules remaining in the set.

4.1.2. Control

Two inference mechanisms are prevalent in rule based languages: forward chaining and backward chaining. Forward chaining uses the state of the data to perform actions leading to a goal state [Maher 86]. Backward chaining involves choosing a goal state, checking to see if the current state of data can satisfy the goal, and if not, creates subgoal states and iterates. [Maher 86] OPS5 uses a forward chaining inference mechanism. Within OPS5, one can write *data-driven* or *goal-driven* rule based systems. Data-driven systems rely completely on the existence of data items in working memory to control rule firings. A goal-driven system uses specific working memory classes, popularly called "goal" wme classes, to control rule firings.

Sometimes it is undesirable for an engineering expert system to be controlled by the state of the data in working memory. If there is a particular process that requires data, it is preferred to use goals to obtain only the data needed instead of trying to provide all possible data. For this reason, many engineering expert systems fit comfortably to goal-driven approaches.

4.1.3. Additional Functions

In order to make OPS5 more user friendly and functionally more capable, extra functions were written in the Civil Engineering Department at Carnegie Mellon University to run external to OPS5: Fill and math. Fill enables a more user friendly and robust method of user input. The math functions define a set of arithmetic functions for more powerful computational capabilities.

4.1.3.1. Fill Function

Fill is a function that can be called from the RHS to query the user for the attributes of a working memory element (wme); and do so with type checking and robustness. It is named Fill because it fills the value fields of the attributes of a working memory element and assigns a value of "filled" to the "status" attribute of the wme. Before calling Fill, a set of *putprop* statements must be written containing the list of questions to be asked about the attributes, and the type of answer the user is supposed to return for each attribute. There is a choice of real, integer, number, string, boolean, and multiple choice menu types. If the user specifies anything other than the type specified in the putprops for that attribute, Fill will print a message and let the user try again. Fill requires approximately 30 k-bytes of memory.

4.1.3.2. Math Function

The math functions are a set of defined lisp functions that expand the math capabilities of OPS5, and can be called from the RHS to perform trigonometric, logarithmic, comparison, and exponential functions. The math functions require approximately 10 k-bytes of memory.

4.2. Software

The lisp environment in which OPS5 runs is Golden Common Lisp, a product of Gold Hill Computers, Cambridge, Massachusetts. The version of GCL used for this OPS5 is version 2.1, an extended memory version requiring 512K of RAM with at least 1 megabyte of extended memory.

4.3. Hardware

The software runs on an IBM PC-AT, with 640 k-bytes of RAM, 3 megabytes of extended memory, and runs at 6MHtz. The Intel 80286 micro processor along with a 16 bit bus and 44 megabyte hard disk drive give this machine adequate speed for load up and execution of the GC-Lisp environment, OPS5 programming language, and SEES code.

Chapter Five

Knowledge Organization

The knowledge in SEES is composed of declarative knowledge stored in OPS5 working memory and production rule knowledge stored in production memory. This chapter describes both kinds of knowledge in detail.

5.1. Declarative Knowledge

Declarative knowledge, stored in OPS5 working memory, is either permanent domain knowledge or problem specific knowledge generated during execution.

5.1.1. Domain Declarative Knowledge

Domain knowledge is tabular working memory knowledge that is always available during execution. In SEES, domain knowledge consists of historical and current knowledge about engineering design parameter values, providing a permanent database.

Examples of working memory elements for storing the strength reduction factors used in concrete design, as per section 9.3.2 of the ACI code, is shown below.

```
(strength_reduc_factor ^application flexure
                       ^phi 0.90            )
(strength_reduc_factor ^application tension
                       ^phi 0.90            )
(strength_reduc_factor ^application compression
                       ^phi 0.70            )
(strength_reduc_factor ^application shear_torsion
                       ^phi 0.85            )
(strength_reduc_factor ^application bearing
                       ^phi 0.70            )
```

The other domain knowledge working memory classes are:

- min_beam_thickness - table 9-5a of ACI 318-83
- steel_history - changes in yield strength with time
- concrete_history - changes in yield strength with time
- cover_history - cover requirements with time
- rebar_specs - numbers and size information

The attributes and values of these classes are given in Appendix A.

5.1.2. Problem Specific Declarative Knowledge

Problem specific knowledge is represented as working memory elements that have different values for each execution. This type of knowledge is used in the following ways.

goal class knowledge	control information
problem specification knowledge	problem formulation
design parameter knowledge	engineering design information
evaluation criteria knowledge	evaluation criteria information

5.1.2.1. Goal Class Knowledge

The goal working memory class, as shown below, serves as the basis for controlling the solution process. A goal wme is the first premise of every rule, using the MEA conflict resolution strategy to consider the most recent goal and task.

```
(goal
      name          ;the name of the goal
      task          ;what the goal may be interested
                    ;in particular
      relative_to   ;the consideration under which
                    ;the goal with that
                    ;task is interested (optional).
                    ;may be "i" if immaterial
      status        ;active or complete
)
```

The attributes of a goal working memory element play a role in further determining the

applicability of a given rule. The *task* attribute allows multiple instances of goals with identical names. For example, multiple instances of the *get_parameters* goal are distinguished by tasks corresponding to the names of the individual parameters. The *relative_to* attribute further discriminates by specifying the material or shape of the element. For example, when there are goals named *get_parameters* with tasks called *d* (depth of member), distinction between acquisition of d for steel beams and d for concrete beams is provided by the *relative_to* attribute. The status attribute indicates whether a goal is active or completed.

5.1.2.2. Problem Specification Knowledge

Problem specification knowledge is information obtained from the user during problem formulation. This information is stored in three classes of working memory elements: member information, problem definition, and problem criteria, as shown below. Each of these classes has only one instance in a given execution.

```
(member_info
     type      ;either a beam or column
     material  ;either steel or reinforced
               ;concrete or composite or timber
     span      ;either fixed, simple, hybrid, cantilever
     status    ;empty or filled
)

(problem_def
     question_of ; flexure, shear, axial,
                 ;torsion, beam-column
     location    ;midspan, at_ends
     status      ;empty or filled
)

(problem_criteria
     loading     ;seismic or typical in service
     exposure    ;indoor or outdoor
     status      ;empty or filled
)
```

The member_info wme stores the information about the member. The *type* attribute indicates the member is either a beam or column. The *material* attribute indicates the member is made either of steel, reinforced concrete, composite construction, or timber. The *span* attribute defines whether the member is fixed at both ends, simply supported, a hybrid, or cantilever. The problem_def wme contains information on the capacity being considered. The *question_of* attribute defines which of the five capacity parameters are being evaluated and the *location* attribute defines where on the member to evaluate the capacity. The problem_criteria wme stores further problem information. The *loading* attribute defines whether analysis is for normal

in-service or seismic loading and the *exposure* attribute indicates whether the member is indoors or outdoors. All three wme classes have a *status* attribute which indicate when the wme has been filled with values.

5.1.2.3. Design Parameter Knowledge

Design parameter knowledge contains all information about design parameters used in a specific problem solution. Design parameters are represented in three working memory classes: parameter_information, parameter_query, and parameter_value. The parameter_information class contains six attributes: name, inferable, criteria, inferred, method, and over-ride, as described below. The parameter_query class is used by the Fill function to determine if the user can provide the parameter_value. The parameter_value class stores the actual value of a parameter once it is obtained.

```
(parameter_information
        name          ;a parameter name
        inferable     ;yes or no
        criteria      ;for which the parameter
                      ;may be inferred. (eg. year)Could
                      ;be "i" if this is immaterial
        inferred      ;yes or no, Not all infer parameters
                      ; are always inferred.
        method        ;compute or immaterial, some
                      ;parameters are computed
                      ;it is not practical to ask
                      ;the user for them.
        over_ride     ;y or n Whether the user wants
                      ;to override an inferred value.
)
(parameter_query
        name           ;a parameter name
        user_knows     ;yes or no
        status         ;empty or filled
)
(parameter_value
        name           ;a parameter name
        value          ;a number
        status         ;empty or filled
)
```

The parameter_information wme class does the book keeping on a parameter. The *name* attribute stores the name of the parameter. The *inferable* attribute indicates if a parameter can be inferred. If a parameter can be inferred, the *criteria* attribute defines what other parameter is needed to infer the parameter. The *inferred* attribute indicates if a parameter has been inferred.

The *over-ride* attribute indicates if the user wishes to override the value of an inferred parameter. The *method* attribute indicates if a parameter can be computed.

The parameter_query wme class stores the information on the user knowledge of a parameter. The *name* attribute stores the parameter name, the *user_knows* attribute indicates if the user knows the value of the parameter, and the *status* attribute indicates if the element has been filled.

The parameter_value wme class contains the actual value of the parameter once it has been obtained. The *name* attribute stores the parameter name, the *value* attribute stores the value, and the *status* attribute indicates if the parameter has a value.

Vector-attribute working memory classes are used during parameter identification to store lists of the relevant parameters and initialization values. Five lists are used: parameter names, problem types, parameter infer truths, inferring criteria, and acquisition methods. The wme classes are shown below.

```
(relevant_parameters
        param_list          ;the list of relevant parameter names
)

(problem_relations
        param_list          ;problem type associated with relevant
                            ;parameters
)

(param_infer_truths
        param_list          ;information (yes or no) associated
                            ;with whether relevant parameters can be inferred
)

(param_infer_criteria
        param_list          ;information needed to infer values for
                            ;relevant parameters
)

(param_acquire_methods
        param_list          ;method used to obtain parameter value;
                            ;will either have a value of i for immaterial
                            ;when the normal rules for parameter acquisition
                            ;should be used, or a value of compute when the
                            ;parameter should always be computed
)
```

The relevant_parameters wme class stores the *param_list* of relevant parameter names. The problem_relations wme class stores the problem types that the parameter names are tied to. The param_infer_truths and param_infer_criteria wme classes store information on if a parameter can be inferred and what is needed to infer the parameter. The parameter_acquire_methods list stores the information needed to fill the method field of the parameter_information wme class.

5.1.2.4. Evaluation Criteria Knowledge

Two classes of wme contain the evaluation knowledge: eval_criteria and crit_ingredient. The eval_criteria stores information on each of the relevant evaluation criterion for the problem. The crit_ingredient class stores information on the ingredients of each evaluation criterion. The eval_criteria and crit_ingredient wme classes are shown below.

```
(eval_criteria
        name                ;the name of the evaluation criterion
                            ;eg. moment_capacity
        relative_to         ;the problem it is specific to,
                            ;eg. reinforced concrete
                            ;or could be immaterial.
        supposed_greater    ;the criteria test ingredient
                            ;which must be greater in order for
                            ;the test to be satisfied
                            ;eg. member moment capacity, ΦMn
        greater_method      ;method of acquisition of the supposed
                            ;greater ingredient,either compute or
                            ;not_compute
        supposed_lesser     ;the test ingredient that must be
                            ;lesser for the test to be satisfied.
                            ;eg. the applied moment, Mu
        lesser_method       ;either compute or not_compute
        result              ;the result of the test, satisfied or
                            ;not_satisfied
        governing           ;whether this test governs or not
                            ;(yes or no)
        displayed           ;flag for the print results goal
                            ;(yes or no)
        status              ;empty or filled
)
(crit_ingredient
        name            ;the name of the ingredient.
                        ;eg. Mu
        relative_to     ;the problem it is specific to,
                        ;eg. reinforced_concrete
        method          ;compute or not_compute - the method by
                        ;which this ingredient can be obtained.
        criteria        ;criteria that this is an ingredient of.
                        ;eg. moment_capacity
        value           ;the value of the ingredient.
                        ; eg. 450,000
        status          ;empty or filled.
)
```

The attributes of the eval_criteria wme class are numerous. The *name* attribute stores the name of the criteria. The *relative_to* attribute stores the problem type of the criteria. The *supposed_greater* and *supposed_lesser* attributes identify the parameters needed for the test.

The *supposed_greater_method* and *supposed_lesser_method* indicate whether the criteria parameters should be computed or not_computed. When the method is specified as not_compute, the system uses get_parameter rules from the redesign goal to obtain the parameter value. The *result* attribute holds the result of the test as satisfied or not_satisfied. The *governing* attribute indicates if the test governs over others. The *displayed* attribute indicates if the print_results goal has echoed the test result yet. Finally, the *status* attribute field indicates if the eval_criteria has been tested yet.

The attributes of the crit_ingredient wme class contain information that ties them to their associated criteria and stores their values. The *name* attribute stores the name of the parameter. The *relative_to* attribute stores the problem type that the parameter is related to. The *method* attribute indicates whether the parameter is computed or obtained through the get_parameter rules in the redesign goal. The *criteria* attribute ties the parameter to its associated criteria. The *value* attribute stores the value of the parameter. Finally, the *status* attribute indicates whether the value has been obtained.

5.2. Knowledge in Rules

Production rules are used to represent process knowledge and problem solving knowledge. Process knowledge controls the solution by making goals. Problem solving knowledge includes knowledge of relevant parameters and criteria for specific problem types.

It is important to divide related rules into separate groups, because of the size of the complete SEES system. Figure 5-1 shows the division of the SEES rules into separate groups for identification purposes. The figure shows the division of process and problem solving rules. All process rules reside in a single group called *control*. Problem formulation rules exist in a single group in the problem solving rules. General rules for redesign and evaluation exist in one group. Special rules for obtaining particular parameter values in redesign are in the getparam group. Rules for redesign and evaluation specific to the problem type are grouped into sets corresponding to the eight problem types. The rule group names as shown in Figure 5-1 are referred to when explaining the three kinds of SEES rule knowledge.

5.2.1. Process Knowledge

The process knowledge in SEES is applicable to all problem types. The process knowledge controls the goal formulation and satisfaction. The control goal tree for SEES is shown in Figure 5-2. At the top level are two goals: evaluate_structural_member and print_results. The evaluate_structural_member goal represents the essence of SEES. This goal is further decomposed into three goals: formulate, redesign, and evaluate. The formulate goal is decomposed into three lower level goals: get_member_type, get_capacity_name, and

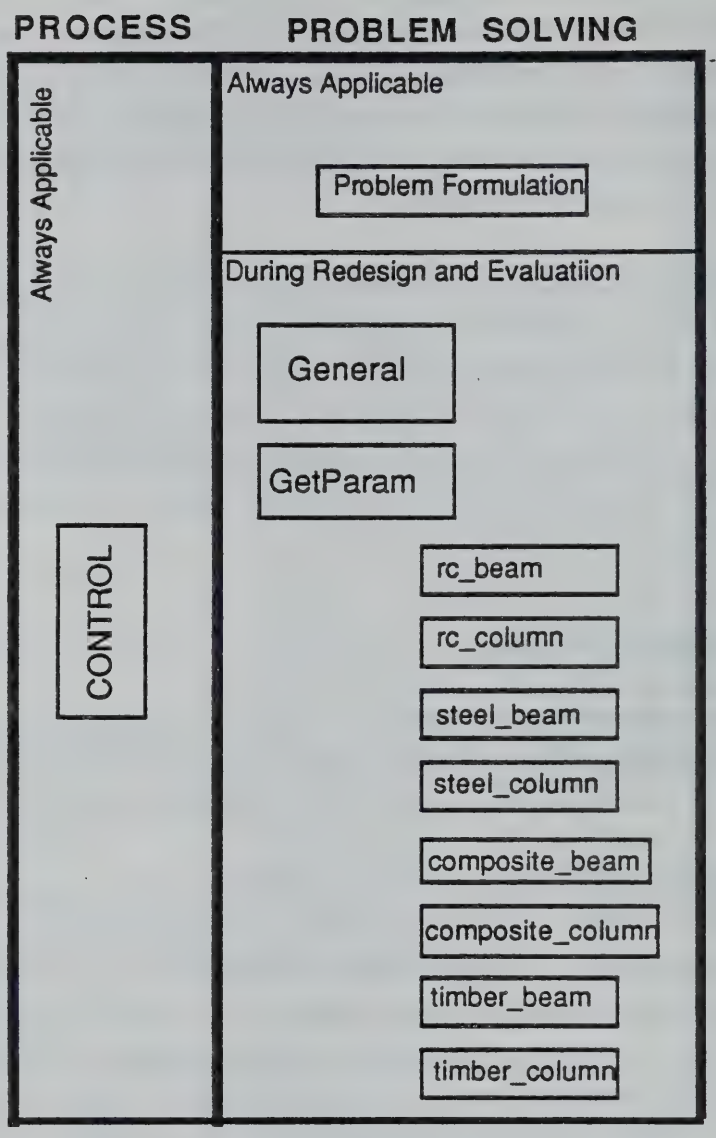

Figure 5-1: Grouping of SEES Production Rules

get_problem_criteria. The redesign goal is decomposed into the identify_relevant_parameters goal and the get_parameter_values goal. The evaluate goal is decomposed into the identify_relevant_criteria, fill_eval_criteria_ingredients, and evaluate_tests goals. Of these three, the fill_eval_criteria_ingredients goal is further decomposed into the identify_ingredients and get_ingredients goals.

The next paragraphs explain the process knowledge of the goal tree in detail. The top level is discussed, followed by formulation, redesign, and evaluation.

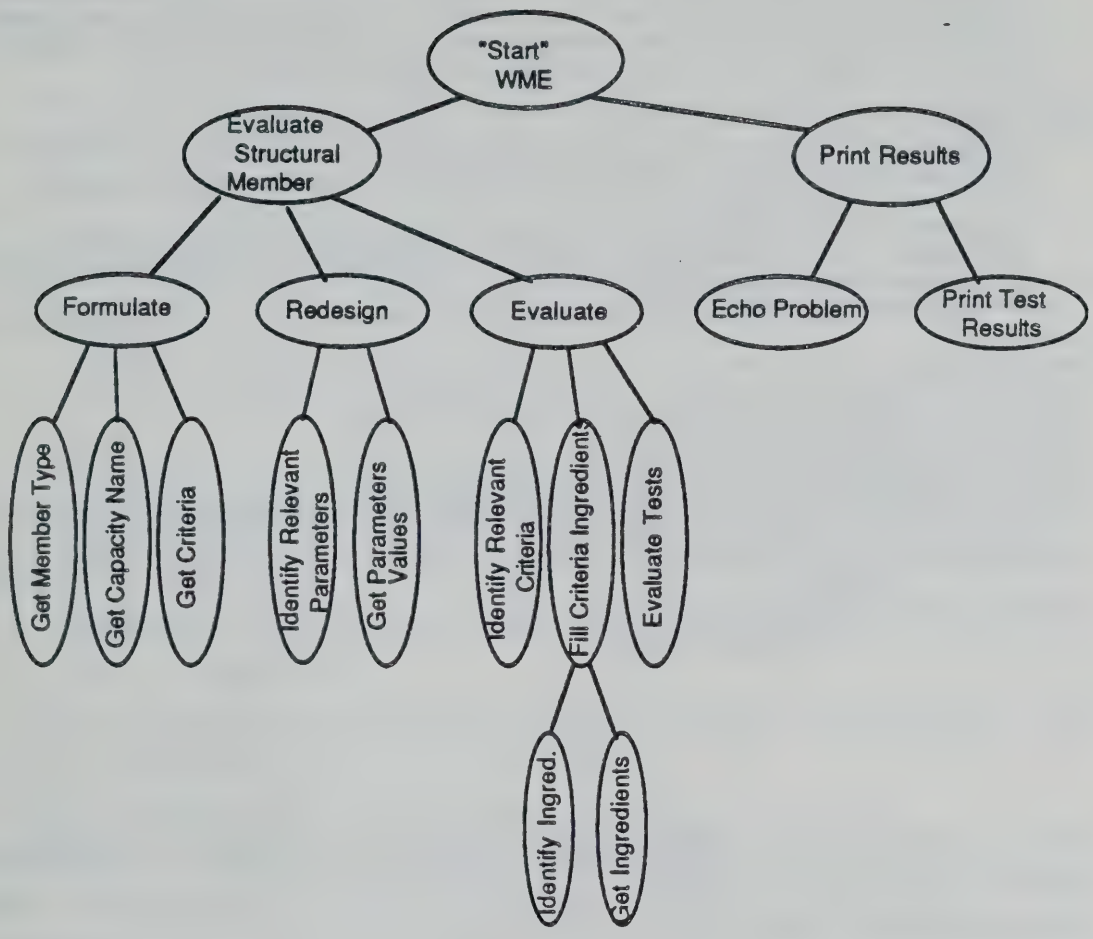

Figure 5-2: SEES Control Tree

5.2.1.1. The Top Level

The "start" working memory element causes the first rule to fire. This rule is shown below.

```
(p init_top_level_goals_and_print_header
 ;*****************************************************************************
 ;  Abstract: The following rule sets active the two top level system goals to
 ;  evaluate the member and print the results.  It also makes the top level
 ;  problem information working memory elements that will hold the global
 ;  problem information.  This rule also prints the program header.
 ;
 ;  Rule:
 ;  IF start exists
 ;  THEN make the top level problem formulation working memory elements
 ;       with status empty
 ;  AND make a goal to evaluate the structural integrity of the member
 ;  AND make a goal to print the results of the analysis with status active
 ;  AND write the problem header
 ;*****************************************************************************
(start)
-->
(make goal ^name print_results ^status active)
(make goal ^name eval_struc_member ^status active)
(owrite(crlf)(crlf)**Greetings**)
(owrite(crlf)This is the second prototype of -SEES-)
(owrite(crlf)(crlf)-- a knowledge based expert system named after chapter 20
                    of the ACI code)
(owrite(crlf)that helps in the evaluation of strength of in situ structural
            members)
(owrite(crlf)(crlf)This current prototypes capability is - reinforced
                  concrete beams-)
)
```

This rule makes two goals: evaluate_structural_member and print_results. OPS5 will always perform the goal made last because that goal has the most recent time tag. Since the evaluate_structural_member goal was mentioned last, it takes precedence. The next rule to fire creates the three subproblem goals underneath the evaluate_structural_member goal. They are formulate_prob, redesign, and evaluate. The rule is shown below.

```
(p evaluate_structural_member_goal
 ;*****************************************************************************
 ;  Abstract:  This rule sets the three major goals that are responsible for
 ;  accomplishing the eval_struc_member goal.
 ;
 ;  Rule:
 ;  IF there is an active goal named eval_struc_member
 ;  THEN make the goal to formulate the problem with status active
 ;  AND make the goal to redesign with status active
 ;  AND make the goal to perform the evaluation with status active
 ;
 ;*****************************************************************************
(goal ^name eval_struc_member ^status active)
-->
(make goal ^name perform_evaluation ^status active)
(make goal ^name redesign ^status active)
(make goal ^name formulate_problem ^status active)
)
```

The remainder of the section describes how each of the three evaluate_structural_member subgoals are satisfied.

5.2.1.2. Problem Formulation

Three lower level goals are created under the problem formulation goal by the following rule. The three goals made are: get_member_type, get_capacity_name, and get_criteria.

```
(p do_formulate_problem_goal
 ;*****************************************************************************
 ;
 ;  Abstract:
 ;     This rule sets the lower level goals that see to the completion of the
 ;  problem formulation goal.
 ;
 ;  Rule:
 ;  IF there is an active goal to formulate the problem
 ;  THEN make the goal to get the member type with status active
 ;  AND make the goal to get the desired capacity name with status active
 ;  AND make the goal to get the problem criteria with status active
 ;
 ;  Source:  Joe Peters
 ;
 ;  Date:    Summer 1987
 ;
 ;*****************************************************************************
(goal ^name formulate_problem ^status active)
-->
(make goal ^name get_criteria ^status active)
(make goal ^name get_capacity_name ^status active)
(make goal ^name get_member_type ^status active)
)
```

The get_member_type goal determines what the member type and material are. The get_desired_strength_parameter goal determines the capacity to be evaluated. The get_criteria goal gets the loading criteria and environmental conditions about the member. Satisfaction of these three goals enables the system control to move to the redesign branch of the control tree.

5.2.1.3. Redesign

The redesign problem is decomposed into the identify_relevant_parameters and get_parameter_values goals as shown in the rule below.

```
(p do_redesign_member_goal
 ;*****************************************************************************
 ;  Abstract: This rule makes those goals necessary to complete the goal to
 ;  redesign the member.
 ;
 ;  Rule:
 ;  IF there is an active goal to redesign
 ;  THEN make the goal to identify the relevant parameters with status active
 ;  AND make the goal to get the parameters with status active
 ;
 ;*****************************************************************************
(goal ^name redesign ^status active)
-->
(make goal ^name get_parameter_values ^status active)
(make goal ^name ident_rel_param ^status active)
)
```

Problem specific rules satisfy these two goals. The identify_relevant_parameters goal determines the list of relevant parameters according to the problem specification. The get_parameter_values goal sets the get_parameters goals for each of the relevant parameters in the list. When the acquisition of relevant design parameters is complete, the system performs the evaluate goal.

5.2.1.4. Evaluation

The rule shown below decomposes the evaluate goal into the identify_relevant_criteria, fill_eval_criteria_ingredients,and evaluate_tests goals.

```
(p do_perform_evaluation_goal
 ;*****************************************************************************
 ;  Abstract:  This rule sets the goals necessary to complete the perform
 ;  evaluation goal ; that goal that looks at the computational results and
 ;  the criteria and comes up with a judgment about the member.
 ;
 ;  Rule:
 ;  IF there is an active goal to perform the evaluation
 ;  THEN make the goal to compute the value of the capacity with status active
 ;  AND make the goal to modify the results with status active
 ;  AND make the goal to make the judgment with status active
 ;
 ;*****************************************************************************
(goal ^name perform_evaluation ^status active)
-->
(make goal ^name evaluate_test ^status active)
(make goal ^name fill_eval_criteria_ingredients ^status active)
(make goal ^name identify_eval_criteria ^status active)
)
```

The identify_criteria goal identifies the criteria for the strength parameter being considered. The

fill_eval_criteria_ingredients goal identifies and gets the values of the ingredients needed for the tests associated with the criteria. This goal is much like the redesign goal because it too must go through a phase of identifying test ingredients and getting the values. The fill_eval_criteria_ingredients goal is decomposed, as shown below, to the identify_criteria_ingredients and get_ingredients goals.

```
(p make_goals_to_ident_and_get_ingred_for_each_criteria
 ;*****************************************************************************
 ;  Abstract:  This rule creates the identify ingredients  and get ingredients
 ;  goals for each of the test criteria.
 ;
 ;  Rule:
 ;  IF there is an active goal to fill evaluation criteria for a task of <name>
 ;       that is relative to a problem
 ;  THEN make the goal to get_ingredients with task <name> with the same
 ;       relation
 ;  AND make the goal to ident_eval_ingred with task <name> with the same
 ;       relation
 ;*****************************************************************************
(goal ^name fill_eval_criteria ^task <name> ^relative_to <relation>
      ^status active)
-->
(make goal ^name get_ingredients ^task <name> ^relative_to <relation>
           ^status active)
(make goal ^name ident_crit_ingredients ^task <name> ^relative_to <relation>
           ^status active)
)
```

When the ingredients of a criteria test have been obtained, the evaluate_test goal performs the actual boolean tests. When all tests have been made for their corresponding criteria, the print_results goal echoes the problem specification information and prints the results of the tests.

5.2.2. Problem Solving Knowledge

This section presents the problem solving knowledge for completing the goals created by the control rules. Presentation of the specific domain knowledge rules required to solve all possible problem types is beyond the scope of this section, so only the general aspects of the problem solving knowledge is discussed. Some of the specific domain knowledge for the solution of reinforced concrete beam problems will be discussed in Chapter 6.

Control of the system is based on a general high level control, as in Figure 5-2, with specific problem control residing in low level rules. With the aid of the working memory objects storing problem specification information, specific rules at the lower levels fire depending on specific cases. For example, during parameter acquisition, there may be a general rule to get steel yield stress. For a steel beam, this rule will fire, while a rule asking for concrete yield stress will not

fire. For a concrete beam problem, both rules would fire since both the steel yield and concrete yield must be known. Some rules fire for more than one problem definition and some rules only fire for specific problem definitions. The remainder of this section describes the general aspects of the low level problem solving knowledge, starting with problem formulation, then redesign, and finishing with evaluation.

5.2.2.1. Problem Formulation

The problem formulation rules query the user for the problem specification information. The Fill function is used in these rules to fill in the wme's for the member type, the desired capacity parameter, and the special considerations of the problem. Constraint rules verify the user input for valid problem specification information. One rule checks for the completion of the three formulation goals, and passes control over to the redesign goal.

5.2.2.2. Redesign

The redesign rules identify and obtain the design parameter values needed to check the structural member strength criteria. The approach taken for this subproblem is that relevant parameters are identified, and separate "get_parameters" goals are made with names of parameters as task attributes. Either general rules for parameter acquisition or specific rules related to a special parameters infer the parameter values. The methods used for parameter identification and acquisition are discussed in the following paragraphs.

Member Shape Information

The member shape must be known before relevant parameters can be identified. Acquisition of this information is not part of problem formulation because relevant member shapes are specific to the problem type. Therefore, the rules to determine the shape of the member are part of the domain rules for the different problem types.

Parameter Identification

The list of relevant parameters for each problem depend on the problem specification. For example the list of parameters for the shear capacity of a reinforced concrete column is different than the list of parameters needed for the axial capacity of a steel column. Conversely, there may be parameters that both problems share, like steel yield strength, f_y. To add confusion, parameters for different problem definitions with the same name may have completely different meanings. An example of this is the parameter d, which has a different meaning for a steel beam than for a concrete beam. Finally, the number of parameters returned for each problem definition is variable.

A table is used to summarize the relevant parameters for the different problem definitions, as

	Problem Definition Columns →												
member_type = beam	X	X	X										etc.
member_type = column													
material = steel													
material = concrete	X	X	X										
material = composite													
material = timber													
capacity = axial													
capacity = flexure	X												
capacity = shear		X											
capacity = torsion			X										
capacity = axial-bending													
shape = I													
shape = T		X	X										
shape = L		X	X										
shape = rectangle	X	X											
shape = box													
steel yield fy	X	X											
concrete yield f'c	X	X	X										
web width bw		X	X										
flange width bf													
box width b	X												
depth d	X	X											
Whitney stress block a	X												
span length l	X		X										
flange thickness hf			X										
depth to comp steel d'	X												
stirrup spacing s		X											
area tension steel As	X												
area comp steel A's	X												
flang overhang			X										
member bay spacing			X										
strength reduc factor $\beta 1$	X												
torsion section prop $\Sigma x^2 y$			X										
area shear steel Av		X											
total member heigth h			X										

Figure 5-3: SEES Table for Relevant Parameter Identification

shown in Figure 5-3. Problem specification information is listed in the upper left quadrant of the table. The list of all possible parameters for all problem definitions is the lower left quadrant. Each problem specification represents a single column in the right half of the table where conditions are marked with an "X" and the corresponding set of parameters are also marked. For example, for the shear capacity of either a rectangular, T, or L shaped reinforced concrete beam, the steel yield, f_y, concrete yield, f'_c, web width, b_w, member depth, d, stirrup spacing,s and shear steel area, A_v are relevant.

One OPS5 rule is used for each column of the table. Each rule returns a list of the relevant parameters to the vector-attribute working memory element called *relevant_parameters*. Each rule also returns an associated list of problem types corresponding to the parameter list. These relations resolve the confusion resulting from parameters in different problems having the same name but dissimilar meaning. Also returned in each rule are two lists that indicate if a parameter can be inferred and give the inferring criteria information about the parameters. These lists are used to initialize the parameter_information working memory elements for each parameter. Finally, a list is returned which specifies the acquisition method for parameters that are not to be obtained through generic means. The identification rule also passes a single valued working memory element called rel_param_index of the number of parameters in the list. An example of a rule identifying the relevant parameters for checking flexure of a rectangular reinforced concrete beam follows:

```
(p ident_param_for_rectangular_flexure
 ;*****************************************************************************
 ;
 ;  Abstract:  This rule identifies the parameters that are relevant to
 ;  solution of the flexural capacity of a rectangular reinforced concrete
 ;  beam.
 ;
 ;  Rule:
 ;  IF there is an active goal to identify relevant parameters
 ;  AND there are no relevant parameters yet
 ;  AND the problem definition shows a desired capacity of flexure
 ;  AND the member info shows the beam type with RC_concrete material
 ;  AND the shape info shows a rectangle
 ;  THEN make the wme relevant_parameters with vector attribute param_list
 ;       to contain b, d, d_prime, fpc, fy, As, A_prime_s, beta_one, a, 1
 ;  AND make the list of problem relations
 ;  AND make the list of infer truths
 ;  AND make the list of infer criteria
 ;  AND make the list of acquisition methods
 ;  AND make the wme rel_param_index with an index of 10
 ;  AND make the double_query wme with status empty
 ;  AND modify the identify goal to status complete
 ;*****************************************************************************
```

```
((goal ^name ident_rel_param ^status active)<goal>)
-(relevant_parameters)
(problem_def ^question_of flexure)
(member_info ^type beam ^material RC_concrete)
(shape_info ^shape rectangle)
-->
(make relevant_parameters ^param_list b d f_prime_c fy As
                                      beta_one a l A_prime_s d_prime)
(make problem_relations ^param_list i rc i rc i i rc i rc_double rc_double)
(make param_infer_truths ^param_list no yes yes yes yes yes yes no no no)
(make param_infer_criteria ^param_list i i year year i i i i i i)
(make param_acquire_methods ^param_list i i i i compute i compute i ask_only
                                        ask_only)
(make rel_param_index ^index 10)
(make double_query ^status empty)
(modify <goal> ^status complete)
)
```

This rule returns a list of ten parameters for flexural analysis of rectangular reinforced concrete beams. The relevant problem types are rc (reinforced concrete) or rc_double (indicating a double reinforcing parameter). All of the parameters can be inferred except for the width of the member, b. Two parameters are inferred by the year of construction: the concrete yield strength f'_c and the steel yield strength f_y. Two parameters are automatically computable: A_s and *a*, and the two double reinforcing parameters can only be provided by the user ("ask_only").

Once the list of relevant parameters and the associated lists of properties have been returned, they are decomposed to establish their individual instances in working memory. A single iterative rule in the general rule group takes the parameters off the list one at a time using the rel_param_index as a counter, and creates the *parameter_information, parameter_query,* and *parameter_value* working memory elements for that parameter. The iterative rule also creates the get_parameters goals with tasks equaling the parameter name.

Getting Parameter Values

There are three ways to get the values of the identified parameters:

1. User query, where the user is asked directly of his knowledge for the particular parameter.
2. Computation of the parameter using other parameters that are already known.
3. Inference using the heuristics in the knowledge base of the system.

For most parameters, except those specifically computed, the user is always asked first for the value of the parameter. If the user does not know a parameter value, the system computes or infers the value.

Two types of rules comprise parameter acquisition: general and specific. The general rules perform operations that apply to all parameters. These rules perform the following:

1. Ask user if he knows a parameter value.
2. Ask for the value of a parameter if the user knows it.
3. Warn the user that the system cannot proceed if the user does not know a parameter that cannot be inferred.
4. Use Fill for a parameter value if the user wishes to proceed after the above warning.
5. Set the goal to compute a parameter for those parameters that are to be computed and not queried for.
6. Set the *get_parameter* goal status to complete for those parameters that are computed.
7. Set the goal to infer the parameter if the user does not know it and it can be inferred.
8. Check for the user satisfaction of an inferred parameter.
9. Refire Fill if the user is unsatisfied with an inferred parameter.
10. Make the goal to get the year of construction if it is needed to infer a parameter.

Copies of these rules can be found in Appendix A.

Specific rules determine parameter values when goals have been made to infer or compute them. Below is an example of a rule that infers the value of the area of tension steel, A_s for reinforced concrete flexure problems. This rule fires after previous attempts to get the area of steel have failed. It gets the steel area by taking one percent of the member cross sectional area, assuming a one percent reinforcing steel ratio.

```
(p infer_area_of_steel_to_be_one_percent
 ;*****************************************************************************
 ;  Abstract:  This rule performs the last acquisition of As after all previous
 ;  attempts fail.  If the user did not know the bar specs, and was not able
 ;  to answer to the steel area directly, then this rule computes the value of
 ;  the steel are by assuming a one percent reinforcement ratio.  This is
 ;  a reasonable guess.
 ;
 ;  Rule:
 ;  IF there is an active goal to infer_parameters with task As
 ;  AND As is not known yet
 ;  AND there is parameter_information on the parameter
 ;  AND the member width b or bw is known
 ;  AND the member depth d is known
 ;  THEN compute the area of steel as being one percent of the member area
 ;  AND give the result to As
 ;  AND write what you just did
 ;  AND modify the parameter_information to inferred - yes
 ;  AND complete the goal
 ;*****************************************************************************
{(goal ^name infer_parameters ^task As ^status active)<goal>}
{(parameter_value ^name As ^value nil ^status empty)<parameter_value>}
{(parameter_information ^name As)<parameter_information>}    ;for binding
(parameter_value ^name << b bw >> ^value <b> ^status filled)
(parameter_value ^name d ^value <d> ^status filled)
-->
(bind <result> (compute 0.01 * <b> * <d> ))
(modify <parameter_value> ^value <result> ^status filled)
(owrite (crlf)(crlf)User unable to specify area of steel information)
(owrite (crlf)**System will assume existence of at least one percent steel in
                member)
(owrite (crlf)Area of tension steel used computed from web width of <b> inches
              and member)
(owrite (crlf)depth of <d> inches)
(owrite (crlf)**One percent yields a steel area of <result> square inches)
(modify <parameter_information> ^inferred yes)
(modify <goal> ^status complete)
)
```

Redesign rules represent a sizable portion of the SEES problem solving rules. When redesign is finished, the system moves into the evaluation goal.

5.2.2.3. Evaluation

Evaluation rules identify the relevant criteria, identify the criteria ingredients, get ingredient values, perform the tests, and report the results to the user. The following paragraphs describe the evaluation steps.

Criteria Test Identification

Identifying criteria is similar to identifying parameters in the redesign goal. Five additional lists are returned with the relevant_criteria list. A problem relations list specifies problem type. A *supposed_lesser* list identifies the ingredient that is I_l in the boolean test $I_l < I_g$. A *supposed_lesser_method* indicates how each of the supposed_lesser ingredients are to be

obtained: compute or not_compute. A *supposed_greater* list identifies I_g in the boolean test. The *supposed_greater_methods* list identifies the supposed greater ingredients' method of acquisition. A rel_criteria_index is returned with the number of criteria in the list. An example of a rule for identifying the evaluation criteria ingredients for evaluating shear capacity of a reinforced concrete beam is shown below.

```
(p identify_general_rc_beam_shear_criteria
 ;*************************************************************************
 ;  Abstract:  This rule fires under the identify eval criteria goal.
 ;  It creates working memory elements with the names of the
 ;  evaluation tests to be made for checking the shear capacity of a
 ;  rc_beam.
 ;
 ;  Rule:
 ;  IF there is an active goal to identify evaluation criteria
 ;  AND the member info shows member type beam with RC_concrete
 ;      material
 ;  AND the problem definition shows a question of shear
 ;  THEN return the list of relevant criteria as stirrup_spacing,
 ;     min_shear_reinforcement, no_stirrups_needed, total_shear,
 ;     raw_concrete_shear,
 ;  AND return the list of corresponding problem relations
 ;  AND return the list of supposed greater and lesser criteria
 ;     ingredients
 ;  AND return the sister lists of ingredient methods of acquisition
 ;  AND make the criteria index equal to five
 ;  AND modify the identification goal to status complete
 ;*************************************************************************
{(goal ^name identify_eval_criteria ^status active)<goal>}
-(relevant_criteria_names)
(member_info ^type beam ^material RC_concrete)
(problem_def ^question_of shear)
(shape_info ^shape <shape> ^status filled)
-->
(make relevant_criteria_names ^criteria_list spacing
    min_shear_reinforcement no_stirrups_needed total_shear
     raw_concrete_shear)
(make criteria_problem_relations ^criteria_list rc rc rc rc rc)
(make supposed_greater_crit_ingredients ^criteria_list
        maximum_stirrup_spacing Av phi_Vc_by_two phi_Vn phi_Vc)
(make supposed_greater_acquire_methods ^criteria_list compute
                                  not_compute compute compute compute)
(make supposed_lesser_crit_ingredients ^criteria_list stirrup_spacing
                                   Av_rqd Vu Vu Vu)
(make supposed_lesser_acquire_methods ^criteria_list not_compute
                         compute not_compute not_compute not_compute)
(make rel_criteria_index ^index 5)
(modify <goal> ^status complete)
)
```

The above rule identifies five evaluation criteria: raw_concrete_shear, total_shear,

no_stirrups_needed, min_shear_reinforcement, and _stirrup_spacing. The list of problem relations all specify reinforced concrete. A total of ten ingredients are identified: ΦV_c, ΦV_n, $\Phi V_c/2$, A_v, maximum_stirrup_spacing, V_u (3 times), A_{vrqd}, and stirrup_spacing, and the methods of acquisition are either *compute* or *not_compute*.

As in the redesign goal, there is a general rule which decomposes the lists one item at a time and creates *fill_eval_criteria* and *evaluate_criteria* goals with task attributes equaling the names of the criteria. The rule also initializes the eval_criteria wme's for each criteria test and inserts the corresponding supposed lesser and greater ingredients into the attributes. The form of this decomposition rule is much like the rule in the redesign goal, so it will not be illustrated here. When the list is decomposed, the ingredients must be identified and obtained.

Ingredient Identification

When criteria identification is complete, another identify-get cycle occurs at the criteria ingredients level. To identify criteria ingredients, the following general rule initializes the crit_ingredient working memory elements for the supposed_lesser and supposed_greater ingredients of each criterion.

```
(p identify_eval_ingredients_for_criteria_test
;*****************************************************************************
;  Abstract:  This is a general rule that makes the crit_ingredient wme's
;  for the relevant evaluation criteria tests for the problem specification.
;  It works in general by checking the existence of a goal to identify the
;  ingredients under a certain task and related to a particular problem type,
;  and then checks for a initialized wme eval_criteria with the same name and
;  problem relation with the supposed greater and lesser quantity names in
;  the test. It then makes the crit_ingredient wme's with the supposed lesser
;  and greater names along with their corresponding methods of acquisition.
;
;  Rule:
;  IF there is a goal to ident_crit_ingredients with task <name> relative to
;      <relation> thats active
;  AND there is an eval_criteria wme with the same name and problem relation
;      with supposed greater and lesser names and methods
;  THEN make the crit_ingredient wme's with the supposed lesser and greater
;       ingredient names and problem relations and methods
;  AND modify the goal to complete
;*****************************************************************************
```

```
{(goal ^name ident_crit_ingredients ^task <name> ^relative_to <relation>
       ^status active)<goal>}
(eval_criteria ^name <name>  ^relative_to <relation>
                             ^supposed_greater <greater>
                             ^greater_method <gmethod>
                             ^supposed_lesser <lesser>
                             ^lesser_method <lmethod>
                             ^status empty)
-->
(make crit_ingredient ^name <greater> ^criteria <name>
                      ^relative_to <relation>
                      ^method <gmethod> ^status empty)
(make crit_ingredient ^name <lesser> ^criteria <name>
                      ^relative_to <relation>
                      ^method <lmethod> ^status empty)
(modify <goal> ^status complete)
)
```

The above rule relies on the existence of a ident_crit_ingredients goal with a given criterion task, and an eval_criteria wme with the same name as the goal's task. It then initializes crit_ingredient working memory elements for both the supposed_lesser and supposed_greater ingredients of the eval_criteria wme, passing the problem relations, criteria names, and methods of acquisition.

The rules that get the ingredient values rely on the existence of the get_ingredients goal for a criterion and a constituent crit_ingredient wme with empty status. An example of a LHS for a rule getting an ingredient value is written below.

```
(goal ^name get_ingredients ^task moment ^status active)
{(crit_ingredient ^name phi_Mn ^value nil ^status empty)<crit_ingredient>}
```

Getting Ingredients

There are three ways in which ingredient values can be obtained:

1. Ingredients that have already been obtained as design parameters in the redesign goal are merely given the value previously obtained.
2. Get_parameter goals are set for ingredients that are obtainable only through user query, like applied loads.
3. Specific rules compute ingredient values using parameters obtained during the redesign goal.

Performing the Tests

After all ingredients have values, the evaluation can be performed. Two generic rules can fire:

the criteria satisfaction rule or the criteria dissatisfaction rule. The criteria dissatisfaction rule, shown below, checks for the "evaluate_criteria" goal, an un-filled eval_criteria wme, and two associated crit_ingredient wme's where the value of the supposed greater ingredient is less than the value of the supposed lesser ingredient. If these conditions are true, the *result* attribute of the eval_criteria wme is given a value of *not_satisfied*.

```
(p check_dis_satisfaction_of_criteria_test
 ;*****************************************************************************
 ;  Abstract:  This is  general rule that checks for the dis-satisfaction
 ;  of the evaluation criteria.  This rule checks the opposite case of that
 ;  checked in the above rule.
 ;
 ;  Rule:
 ;  IF there is an active goal to evaluate_criteria with task <criteria_name>
 ;  AND there is an eval_criteria wme named <criteria_name> that is empty
 ;   with a supposed_greater of <greater> and supposed_lesser of <lesser>
 ;  AND there is a crit_ingredient named <greater> that is filled
 ;  AND there is a crit_ingredient named <lesser> that is filled
 ;  AND <greater> is less than <lesser>
 ;  THEN modify the eval_criteria to filled with a value in result field
 ;         of not_satisfied
 ;  AND modify the goal to complete
 ;  AND write the result
 ;*****************************************************************************
{(goal ^name evaluate_criteria ^task <criteria_name> ^status active)<goal>}
{(eval_criteria ^name <criteria_name> ^supposed_greater <ingred_one>
                                      ^supposed_lesser <ingred_two>
                                      ^result nil
                                      ^status empty)<eval_criteria>}
(crit_ingredient ^name <ingred_two> ^value <hopefully_lesser> ^status filled)
(crit_ingredient ^name <ingred_one>
                 ^value { <hopefully_greater> < <hopefully_lesser> }
                 ^status filled)
-->
(modify <eval_criteria> ^result not_satisfied ^status filled)
(modify <goal> ^status complete)
(owrite (crlf)(crlf)***** Evaluation Criteria Test *****)
(owrite (crlf)<criteria_name> evaluation criteria just tested to see if
                    ingredient)
(owrite(crlf)-- <ingred_one>)
(owrite (crlf)is greater than or equal to ingredient <ingred_two> yields
              result of)
(owrite(crlf)-- not satisfied)
)
```

The criteria satisfaction checking rule is similar to this rule, except it checks to see if the supposed_greater ingredient is greater than the supposed lesser ingredient. When all evaluation criteria have been tested and filled, the print_results goal takes precedence and echoes the problem specification information and prints a summary of all the tests made.

5.3. Summary

Knowledge in SEES is separated into declarative knowledge in OPS5 working memory and rule knowledge in OPS5 production memory. Declarative knowledge is either permanent domain knowledge or dynamic problem specific knowledge. Six wme classes make up the domain declarative knowledge and thirty-four wme classes make up the problem specific declarative knowledge. Knowledge in rules is either process or problem solving knowledge. Process knowledge controls the general solution process and problem solving knowledge provides the detailed solution knowledge. Thirteen rules comprise the process knowledge and sixty-one rules comprise the general problem solving knowledge.

In order to make the system capable of identifying and getting parameters, identifying test criteria and their ingredients, getting ingredient values, and performing the evaluation tests, many rules are needed. When programming the system, generality is used wherever possible to cut down on the number of rules. Whenever there is a particular uniqueness to a parameter, separate rules are needed for its acquisition. When enough parameters share a common attribute, this usually merits addition of an attribute in the general *parameter_information* working memory class and allow new general parameter acquisition rules to fire for that parameter.

Chapter Six
Evaluating Reinforced Concrete Members

The current prototype of SEES addresses one of the eight problem types: reinforced concrete beams. The prototype evaluates flexural, shear, or torsional capacity of reinforced concrete beams. SEES can be easily expanded by adding more heuristics since the implementation separates problem solving rules and problem specific heuristic rules.

This chapter presents an overview of the current prototype of SEES, describing the prototype's extent and scope, and the detailed knowledge incorporated into the knowledge base for evaluation of reinforced concrete beams.

6.1. Problem Overview

This section presents an overview of the current prototype by describing the extent, and the input and output.

6.1.1. Extent

As mentioned previously, the current prototype is capable of evaluating flexural, shear, and torsional capacity of reinforced concrete beams. The system does not evaluate axial or axial-bending capacity of beams.

Three common beam member shapes are considered: rectangular, T, and spandrel (L) sections. Checking these three member shapes for flexure, shear, and torsional capacity results in nine sub-problem types within the reinforced concrete beam problem type. Considering double reinforcement adds to the complexity of the problem since double reinforcement requires more computation than single reinforcement. Figure 6-1 shows a 3X3 grid of the subproblem types for reinforced concrete beams with a brief description of the system capabilities in each element.

The prototype is capable of identifying the relevant parameters and criteria to evaluate flexural capacity for all three member shapes. Moment capacity is computed using moment equations for double reinforcing. If members are, or are assumed to be singly reinforced, the double reinforcement parameters will be assigned zero values during the redesign goal, and should not affect the computation.

CHECKING CAPABILITIES	MEMBER SHAPES: RECTANGLE	MEMBER SHAPES: T	MEMBER SHAPES: SPANDREL (L)
FLEXURE	• Identifies relevant parameters and checks four relevant criteria. • Not capable of checking moment capacity by concrete crushing.	• Identifies relevant parameters and checks four relevant criteria. • Does not compute moment capacity.	• Identifies relevant parameters and checks four relevant criteria. • Does not compute moment capacity.
SHEAR	• Identifies relevant parameters and checks five relevant criteria.	• Identifies relevant parameters and checks five relevant criteria.	• Identifies relevant parameters and checks five relevant criteria.
TORSION	• Identifies relevant parameters and checks one relevant criteria. • Checks base concrete torsional resistance.	• Identifies relevant parameters and checks one relevant criteria. • Checks base concrete torsional resistance.	• Identifies relevant parameters and checks one relevant criteria. • Checks base concrete torsional resistance.

Figure 6-1: Capabilities of Current Prototype on Reinforced Concrete Beams

The system is capable of evaluating shear capacity for all three member shapes. Because of the simplicity of the check, the rule for returning the list of relevant design parameters, the rule for returning the relevant criteria, and the rules for computation and acquisition of criteria ingredients are generic to the three member shapes. Full evaluation capabilities for shear capacity are provided.

The system is also capable of evaluating torsional capacity of all three member shapes. Two rules are needed for relevant parameter identification; one for rectangular sections and one for T and L sections. Only one rule is needed for identification of relevant criteria for all three shapes. Only one criteria, *base member torsional resistance*, is checked for torsion since inference of anything more accurate is very difficult.

The prototype is not yet capable of considering seismic loading or any special condition criteria

like corrosion or concrete spalling. These conditions would be considered during the evaluation subproblem where computed capacities would be reduced to account for the special conditions. This capability has not been incorporated due to lack of expertise as is discussed in Chapter 7. The resulting capability of the prototype is to evaluate capacity of members under normal, in-service conditions.

6.1.2. Input

In order to investigate a reinforced concrete beam problem, the user must specify the information shown in Table 6-1. The member type must be beam, the material must be reinforced_concrete, the capacity desired either flexure, shear, or torsion, the location of the capacity at midspan or ends, the span type either simple, fixed, hybrid, or cantilever, the exposure indoor or outdoor, and the loading in_service.

INPUT ITEM	CURRENT CHOICES
member type	beam
material	reinforced_concrete
capacity desired	flexure, shear, torsion
location of capacity	at_ends or midspan
span type	simple, fixed, hybrid, or cantilever
exposure	indoor or outdoor
loading	in_service

Table 6-1: Prototype Problem Formulation Input

Presently, the system allows the user to specify the member type and material of his choice, but will only react when the above information is specified. Specification of any other information will cause the system to halt due to lack of domain rules to handle those situations.

Within the formulation rules of the prototype is a constraint checking rule that does not allow the user to specify evaluation of axial or axial-bending capacity for beam type members. Checking these two capacities of beams is believed impractical in any case. If the user specifies axial or axial-bending capacity for beams, the system returns a message that an incompatible problem definition has been made, and lets the user re-input new information.

A subset of the input information is not used during the evaluation. Specification of span type, location of load, and loading do not affect the solution in the current prototype because heuristics incorporated in the system do not use that information.

After problem formulation, the user provides member shape information. In order to determine member shape, the user specifies whether the beam is detached from the slab, poured monolithic with the slab, connected with shear connectors, or no slab exists. If monolithic pour or shear connectors are specified, and the beam is interior, the shape is assumed to be T, if the beam is exterior, the shape is L.

The remaining input is concerned with the relevant design parameters for the redesign subproblem and some ingredients in the evaluation subproblem. The user can provide any of the following parameters: concrete and reinforcing steel yield strengths, concrete beam dimensions, rebar sizes, shapes, and quantities, the year of construction, design factors,and beam depth to steel dimensions.

6.1.3. Output

SEES provides the following information to the user.

1. Assumed member shape.
2. Relevant design parameters.
3. Inferred and computed parameters.
4. Relevant evaluation criteria.
5. Source of criteria ingredients: computed, requested, retrieved from parameter attribute.
6. Criteria test results.
7. When evaluation is complete, an echo of the problem specification is written to the user.
8. Summary of problem specification and evaluation.

6.2. Domain Specific Knowledge

This section describes the domain knowledge used to evaluate the capacity of reinforced concrete beams. The knowledge for evaluating flexure, shear, and torsional capacity is provided separately.

6.2.1. Flexure

Some interesting heuristics used in the flexural evaluation are summarized below. They apply to the acquisition of parameters and to decisions made under the evaluate goal and may explain some of the system reasoning mentioned in this section.

- Beams are rarely doubly reinforced, so assume single reinforcement unless the user knows otherwise.
- When the user does not know the area of steel for the system to compute ρ (the reinforcing ratio), 1% is a safe assumption to use for any member.
- If ρ shows signs of being close to or greater than the balanced reinforcement ratio, then failure by concrete crushing should be computed and compared to the failure under normal conditions.
- Even though concrete used long ago did not have as high a yield strength as concrete used today, the fact that concrete gets harder with age justifies the assumption of using 3000 psi yield strength for old concrete (pre 1963). 4000 psi can be used for concrete poured after 1963.
- The yield strength of concrete need not be exact since its effect on the moment equation is about ten percent.
- When inferring steel yield strength, 36,000 psi can be used for construction prior to 1970 and 40,000 psi can be used for construction after 1970.
- Many old beams (pre 1963) are over-reinforced, due to the use of working stress method of design. Under working stress design, the member was always designed to meet working stress requirements. No attention was paid to the mode of failure.
- Deflection is less likely to pose a problem for beams designed after 1963 since enforcement of this criteria has been more strictly imposed with the use of ultimate design practice.
- Using ultimate strength design, the beam will rarely be over-reinforced.

6.2.1.1. Flexure Redesign

The relevant design parameters for evaluating flexural capacity depends on the member shape. Table 6-2 shows the relevant parameters for evaluating flexure of rectangular sections, and T and L sections. The relevant parameters are member dimensions b, d, l, d', b_w, b_f, a, and h_f, material properties f_y and f'_c, reinforcing steel areas A_s, A'_s, and A_{sf} and the factor β_1. The table also shows the parameter problem relations, whether they can be inferred, the criteria for inferring them, and acquisition methods.

Shape	Parameters				
	Name	Problem Type	Inferable	Inferring Criteria	Acquire Meth.
Rect-angle	b	immaterial	no	immaterial	immaterial
	d	rein. concrete	yes	immaterial	immaterial
	f'c	immaterial	yes	year	immaterial
	fy	rein. concrete	yes	year	immaterial
	As	immaterial	yes	immaterial	compute
	β1	immaterial	yes	immaterial	immaterial
	a	rein. concrete	yes	immaterial	compute
	I	immaterial	no	immaterial	immaterial
	A's	rc_double	no	immaterial	ask_only
	d'	rc_double	no	immaterial	ask_only
T -or- L	bw	immaterial	no	immaterial	immaterial
	bf	immaterial	no	immaterial	immaterial
	d	rein. concrete	yes	immaterial	immaterial
	f'c	immaterial	yes	year	immaterial
	fy	rein. concrete	yes	year	immaterial
	hf	rein. concrete	no	immaterial	immaterial
	a	rein. concrete	yes	immaterial	compute
	As	immaterial	yes	immaterial	compute
	Asf	rein. concrete	yes	immaterial	compute
	I	immaterial	no	immaterial	immaterial

Table 6-2: Relevant Parameters for RC Beams in Flexure

The parameter list for evaluating flexural capacity is always a complete parameter list allowing for double reinforcement. Hence, parameters for compression steel areas are returned as well as tension steel areas. Although SEES cannot infer double reinforcement, it allows the user to specify the value of the double reinforcing parameters. If the user does not know the reinforcing details of the beam, SEES assumes single reinforcing and automatically sets the double reinforcement parameters to zero.

As shown in Table 6-2, the parameters b, b_w, b_f, h_f, l, A'_s, and d' must be provided by the user. The methods of inferring the remaining parameters are explained in the following paragraphs.

Steel Yield Strength, f_y

When the user does not know the yield strength of steel, f_y, the year of construction is needed to infer the value from a database of commonly used yield strengths. The values are organized by ranges of years, as shown below.

```
steel_history       start_year 1904
                    end_year 1970
                    fy_min 33000
                    fy_max 50000
                    FS_min 2.06
                    FS_max 2.78
steel_history
                    start_year 1971
                    end_year 1986
                    fy_min 40000
                    fy_max 60000
                    FS_min 2.0
                    FS_max 2.5
```

According to a professional engineer at the NBS, only two division are required for the steel history. For conservatism, only the minimum values are used. The min and max values are left here with safety factors just in case future prototypes wish to incorporate ranges of results.

Concrete Yield Strength

Inferring the yield strength of concrete, f'_c, similar to inferring the steel yield strength, uses a table of values for ranges of years.

```
concrete_history    start_year 1904
                    end_year 1950
                    f_prime_c_min 3000
                    f_prime_c_max 3000
                    FS_min 0.0
                    FS_max 0.0
concrete_history    start_year 1950
                    end_year 1970
                    f_prime_c_min 3500
                    f_prime_c_max 3500
                    FS_min 0.0
                    FS_max 0.0
concrete_history    start_year 1970
```

```
                                    end_year 1987
                                    f_prime_c_min 4000
                                    f_prime_c_max 4000
                                    FS_min 0.0
                                    FS_max 0.0)
```

Only three separations are needed for concrete. As shown, the values used for concrete poured many years ago are higher than expected. The assumption here is that concrete gets harder with age and concrete poured long ago is much harder now than when it was poured. The use of minimum and maximum values is used with factors of safety to allow for the possibility in later prototypes to supply ranges of results. The present prototype uses only the minimum value.

Member Depth to Steel

The member depth to steel, d, is inferred by computation using the lower level parameters: member height, h, concrete cover, and reinforcement bar diameter. The member height must be supplied by the user. If the user does not know the concrete cover, it is inferred using the exposure of the member and the year of construction. The domain knowledge used to infer concrete cover is stored in wme as tables.

```
cover_history       start_year 1904
                    end_year 1920
                    use outdoor
                    value 1.0
cover_history       start_year 1904
                    end_year 1920
                    use indoor
                    value 1.0
cover_history       start_year 1921
                    end_year 1924
                    use outdoor
                    value 2.0
cover_history       start_year 1921
                    end_year 1924
                    use indoor
                    value 2.0
cover_history       start_year 1925
                    end_year 1986
                    use indoor
                    value 1.5
cover_history       start_year 1925
                    end_year 1986
                    use outdoor
                    value 2.0
```

Reinforcement bar diameter is either provided by the user or inferred. If the user does not know the bar size, the default is a number 8 round bar. If the user does know the size and shape, the diameter is derived from a table stored in the knowledge base.

When the member height, cover and rebar diameter are known, d can be computed by the following equation.

$$d = h - cover - \frac{bar\_diameter}{2}$$

Area of Tension Steel

The area of tension steel, A_s, is a computed parameter, using the area of a rebar and the number of bars. If the user knows the number of bars, the area of steel is computed using shape and cross sectional area, otherwise the area of steel is computed as one percent of the member cross sectional area.

Equivalent Stress Block Multiplier β_1

β_1 is inferred using paragraph 10.2.7.3 of ACI 318-83. Three rules in the getparam rule group determine the value of β_1 depending on the value of f'_c as defined by paragraph 10.2.7.3.

$$\beta_1 = 0.85 \quad when \quad f'_c \leq 4000\ psi$$

$$\beta_1 = 0.85 - (\frac{0.2}{4000})(f'_c - 4000) \quad when \quad 4000 < f'_c < 8000\ psi$$

$$\beta_1 = 0.65 \quad when \quad f'_c \geq 8000\ psi$$

Depth of Whitney Stress Block, *a*

The depth of the Whitney stress block, *a*, is a computed parameter depending on the shape of the reinforced concrete beam. For rectangular sections, the equation (equation (4.9a) of [Winter & Nilson 79]) is a function of the area of tension and compression steel, the steel yield strength, the concrete yield strength, and the member width.

$$a = \frac{(A_s - A'_s)f_y}{0.85 f'_c b}$$

For T and L shaped sections the equation (equation (4.19) of [Winter & Nilson 79]) is a function of

the same parameters except the member width is the web width and the area of compression steel is replaced with the flange steel.

$$a = \frac{(A_s - A_{sf})f_y}{0.85f'_c b_w}$$

6.2.1.2. Flexure Evaluation

The relevant criteria for flexure of rc beams shown in Table 6-3, are minimum_steel, where $\rho > \rho_{min}$, ductile_failure, where $\rho < 0.75\rho_b$, moment, where $\Phi M_n > M_u$, and deflection, where $d > d_{rqd}$.

Shape	Criteria					
	Name	Problem Type	Supposed Greater Igredient	Greater Method	Supposed Lesser Igredient	Lesser Method
T, L, Rect-angle	minimum steel	rein. con.	ρ	compute	ρ_{min}	compute
	ductile failure	rc_beam	75% ρ_b	compute	ρ	compute
	moment	rc_beam	Φ Mn	compute	Mu	not com
	deflection	rc_beam	d	not com	d_{min}	compute

Table 6-3: Relevant Criteria for RC Beams in Flexure

Minimum Steel

Satisfying the minimum steel criteria insures that sudden violent failure of a bending member does not occurs when first cracking occurs. To satisfy this criteria, the member reinforcement ratio must be greater than the minimum reinforcement ratio ρ_{min}. The minimum reinforcement ratio is determined using the following equation (10-3 of ACI 318-83).

$$\rho_{min} = \frac{200}{f_y}$$

Ductile Failure

The ductile_failure criteria is satisfied if the reinforcement ratio of the member is less than 75% of the balanced reinforcement ratio, insuring ductile failure of the reinforcing steel occurs before brittle failure of concrete crushing.

The balanced reinforcement ratio depends on the member shape. For rectangular beams, it is computed using the following equation, (4.2) of [Winter & Nilson 79].

$$\rho_b = 0.85\beta_1 \frac{f'_c}{f_y} \frac{87{,}000}{87{,}000 + f_y}$$

The member reinforcement ratio is computed by dividing the area of steel, A_s, by the product of the member depth d and width b. In cases when the area of steel has been inferred to one percent in the redesign goal, the result of this computation comes out to one percent.

Moment

The moment capacity criteria is satisfied if the ultimate beam capacity is greater than the factored applied moment (ultimate strength design). The two ingredients for this test are the factored applied moment M_u and the strength reduced nominal moment capacity ΦM_n. The factored applied moment must be supplied by the user.

The member capacity is obtained by determining a value for the strength reduction factor Φ, computing the nominal moment capacity M_n, and multiplying the two together. The strength reduction factor is determined from a table, shown below, in the domain knowledge database taken from section 9.3 of ACI 318-83. The strength reduction factor is a function of the capacity being checked, and for flexure, equals 0.90.

```
strength_reduc_factor          application flexure
                               phi 0.90
strength_reduc_factor          application tension
                               phi 0.90
strength_reduc_factor          application compression
                               phi 0.70
strength_reduc_factor          application shear_torsion
                               phi 0.85
strength_reduc_factor          application bearing
                               phi 0.70
```

If the ductile_failure criteria is satisfied, the nominal moment capacity is computed. The moment equation (equation (4.15) of [Winter & Nilson 79]) for rectangular beams is shown below.

$$M_n = 0.85 f'_c ab\left(d - \frac{a}{2}\right) + A'_s f_y (d - d')$$

The moment equation (equation (4.21) of [Winter & Nilson 79]) for T and L beams is shown below.

$$M_n = A_{sf} f_y (d - \frac{h_f}{2}) + (A_s - A_{sf}) f_y (d - \frac{a}{2})$$

Deflection

Deflection criteria is satisfied if the depth of the beam is within some prescribed empirical limit. The limitations are defined in Table 9.5(a) of ACI 318-83, and shown below as they are represented in SEES' knowledge base.

min_beam_thickness	span_type simple span_divisor 16
min_beam_thickness	span_type fixed span_divisor 21
min_beam_thickness	span_type hybrid span_divisor 18.5
min_beam_thickness	span_type cantilever span_divisor 8

The table gives the divisors for dividing the span length to provide a minimum depth of beam. All results from the table are multiplied by (0.4 + fy/100000) where f_y is in psi. The two ingredients of the deflection criteria are the member's depth and the required depth as per the table. The depth of the member is a parameter_value working memory element named *d* that has already been filled during the redesign goal. The required depth is a function of the span length and yield strength of the steel, and is computed using the following equation from the footnotes of ACI 318-83 Table 9.5(a).

$$d_{rqd} = (\frac{l}{coeff.})(0.4 + \frac{f_y}{100{,}000})$$

Where the span length is in inches, f_y is in psi, and the "coeff" is the factor from the table.

6.2.2. Shear

As in flexural evaluation, the difficulty in determining shear reinforcement in concrete members lies in not being able to "look" in the concrete to see what is there. In order to evaluate the criteria, the system makes assumptions regarding the shear reinforcing steel, like assuming use of at least number three round bars for stirrups. The uncertainty in inferring shear steel is greater than in inferring flexural steel because design practice of shear reinforcing steel has been much more diverse through the years.

6.2.2.1. Shear Redesign

The relevant design parameters for checking shear of the three member shapes are member dimensions b_w and d, material yield properties f'_c and f_y, and reinforcing parameters A_v and s, as shown in Table 6-4. Also shown are the problem relations, inferring information, inferring criteria, and acquisition methods.

Shape	Parameters				
	Name	Problem Type	Inferable	Inferring Criteria	Acquire Meth.
T	bw	immaterial	no	immaterial	immaterial
-or-	d	rein. concrete	yes	immaterial	immaterial
L	f'c	immaterial	yes	year	immaterial
-or-	Av	rein. concrete	yes	immaterial	compute
Rect-angle	fy	rein. concrete	yes	year	immaterial
	s	rein. concrete	yes	immaterial	immaterial

Table 6-4: Relevant Parameters for RC Beams in Shear

As shown in the table, b_w is the only parameter that is obtained through user query. Here, b_w is used for all three member shapes.

The parameters d, f_y, and f'_c are obtained as they are in checking flexure, and will not be discussed here. The remaining parameters are discussed below.

Area of Shear Steel

Determining the area of shear steel reinforcement, A_v, is very much like determining the value of the tension steel for flexure. If the user is able to supply the stirrup rebar specifications, the area of shear steel is computed as twice the area of the rebar cross sectional area. If the user is unable to supply the rebar specifications, the system assumes use of at least a number 3 round bar and gives a shear steel of 0.22 square inches.

Stirrup Spacing

Stirrup spacing can be supplied by the user or inferred as d/2 where d is member depth. It is questionable whether this is a safe assumption for very old members but is a commonly used assumption.

6.2.2.2. Shear Evaluation

The relevant criteria for evaluating shear capacity of the three member shapes, shown below in Table 6-5, are raw_concrete_shear, where $\Phi V_c > V_u$, total_shear, where $\Phi V_n > V_u$, no_stirrups_needed, where $\Phi V_c/2 > V_u$, min_shear_reinforcement, where $A_v > A_{vrqd}$, and spacing, where $s_{max} > s$.

Shape	Criteria					
	Name	Problem Type	Supposed Greater Ingredient	Greater Method	Supposed Lesser Ingredient	Lesser Method
T -or- L -or- Rect-angle	raw concrete shear	rein. con.	Φ Vc	compute	Vu	not com
	total shear	rein. con.	Φ Vn	compute	Vu	not com
	no stirrups needed	rein. con.	Φ Vc/2	compute	Vu	not com
	min shear rein.	rein. con.	Av	not com	Av rqd.	compute
	spacing	rein. con.	s_{max}	compute	s	not com

Table 6-5: Relevant Criteria for RC Beams in Shear

Raw Concrete Shear

The raw concrete shear criteria is satisfied if $\Phi V_c > V_u$, checking whether the concrete alone can resist the shear forces on the member. This criteria is specified in paragraph 11.3.1.1 of ACI 318-83. The two ingredients of this criteria are the factored applied shear force, V_u, and the strength reduced shear capacity of the concrete ΦV_c. The user must provide the value of the applied load. The value of Φ is obtained as it was for flexure.

The value of the concrete shear capacity is obtained using the following equation (11-3 of ACI 318-83).

$$V_c = 2\sqrt{f'_c}\, b_w d$$

Total Shear

The total shear criteria considers the contribution of the shear reinforcing steel V_s to the

member shear capacity. The criteria is satisfied if $V_u \leq \Phi(V_c+V_s)$, a combination of equation (11-1) and (11-2) in ACI 318-83.

Where Φ, V_c, and V_u are the same as in the raw concrete shear test, and V_s is computed using the equation below((11-17) of ACI 318-83).

$$V_s = \frac{A_v f_y d}{s}$$

No Stirrups Needed

The *no stirrups needed* criteria is satisfied if a minimum amount of shear reinforcement is provided whenever the factored applied shear, V_u, exceeds one half of the raw concrete shear strength ΦV_c (paragraph 11.5.5.1 of ACI 318-83). The two ingredients for this criteria, $\Phi V_c/2$ and V_u, are available from previous criteria checks.

$$V_u \geq \frac{\Phi V_c}{2}$$

Minimum Shear Reinforcement

The *min shear reinforcement* criteria depends on the result of the *no stirrups needed* criteria evaluation. The ingredients of the criteria are the actual shear steel area A_v and the required shear steel area A_{vrqd}. The actual steel area is obtained during redesign, so the system need only recall it. The required steel area is obtained in one of two ways.

1. If the result of the *no stirrups needed* criteria is satisfied, then the required shear steel area is set to zero.
2. If the result of the *no stirrups needed* criteria is not satisfied, then the required steel area is computed using the following equation ((11-14) of ACI 318-83).

$$A_{vrqd} = 50 \frac{b_w s}{f_y}$$

Spacing

The spacing criteria is satisfied if the maximum spacing of the stirrups is not exceeded. The

code requires that stirrup spacing not exceed half the member depth, d. The criteria ingredients are the actual stirrup spacing, s, and the allowable, s_{max}. The allowable is computed as the beam depth over two. The criteria is evaluated as follows.

1. If the user is unable to specify stirrup spacing information in the redesign goal, the system infers a stirrup spacing of d over 2 and the criteria is satisfied.
2. If the user is able to specify a stirrup spacing and it is greater than d over 2 of the member, the criteria is not satisfied.
3. If the user is able to specify a stirrup spacing that is less than d over 2, the criteria is satisfied.

6.2.3. Torsion

6.2.3.1. Torsion Redesign

The relevant design parameters for checking torsion are the member dimensions b, h, l, h_f, b_w, flange_overhang, and member_bay_spacing, the material property f'_c, and the member torsional property, Σx^2y, as shown in Table 6-6.

As the table shows, many of the parameters have already been discussed for flexure and shear. The user must provide b, h, l, h_f, the member_bay_spacing, and b_w. The concrete yield strength, f'_c, is inferred as in flexure and shear. The parameters unique to torsion are the member torsional section property, Σx^2y, and the flange overhang.

Torsional Section Modulus

Torsional section modulus is computed by separating the member cross section into the web rectangle and two overhanging flange rectangles, multiplying the square of the short dimensions to each to the long dimensions and summing the products.

$$\sum_{i=1}^{n} x_i^2 y_i$$

Where *i* is the rectangle number and *n* is the number of component rectangles. Figure 6-2 shows the rectangle component breakdown for T and L sections. Rectangular section are not shown due to their simplicity.

As can be seen by the figure, computation of this parameter depends on knowledge of the member height, web width, slab thickness, and flange overhang.

Shape	Parameters				
	Name	Problem Type	Inferable	Inferring Criteria	Acquire Meth.
Rect-angle	b	rein. concrete	no	immaterial	immaterial
	h	rein. concrete	no	immaterial	immaterial
	f'c	immaterial	yes	year	immaterial
	$\Sigma x^2 y$	rein. concrete	no	immaterial	compute
T -or- L	f'c	immaterial	yes	year	immaterial
	h	rein. concrete	no	immaterial	immaterial
	$\Sigma x^2 y$	rein. concrete	no	immaterial	compute
	overhang	rein. concrete	yes	immaterial	immaterial
	bay spacing	immaterial	no	immaterial	immaterial
	l	immaterial	no	immaterial	immaterial
	hf	rein. concrete	no	immaterial	immaterial
	bw	rein. concrete	no	immaterial	immaterial

Table 6-6: Relevant Parameters for RC Beams in Torsion

Flange Overhang

The flange overhang is a computed parameter following the requirements of section 8.10 of ACI 318-83, as summarized below.

For T Beams

1. Total Flange width should not exceed one quarter of the beam span length.
2. The overhang should not exceed eight times the slab thickness.
3. The overhang should not exceed one-half the clear distance to the next beam.

For L Beams

1. Total flange width should not exceed one quarter of the beam span length.
2. The flange overhang should not exceed one-twelfth of the beam span length.
3. The overhang should not exceed six times the flange thickness.

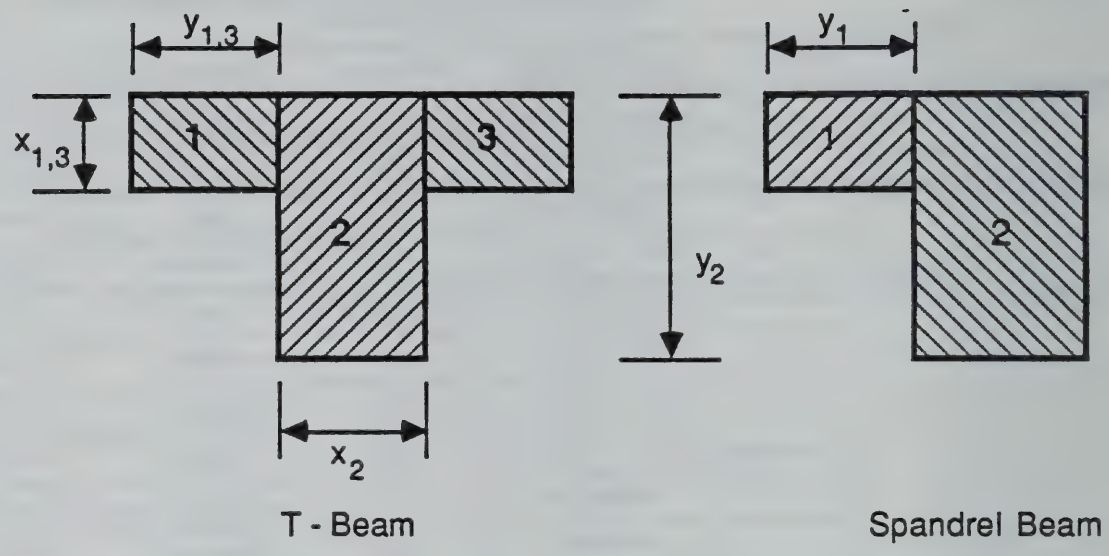

Figure 6-2: Cross Sectional Component Breakdown for T and L RC Beams

4. The flange overhang should not exceed one half the clear distance to the next web.

6.2.3.2. Torsion Evaluation

Evaluating torsional capacity is difficult to consider in isolation because torsion is usually considered in conjunction with shear, and also interacts with flexure because longitudinal reinforcing steel affects the torsional strength of a member. The criterion for evaluating torsional capacity, shown in Table 6-7, is base_torsion_sufficient, where the member_base_torsion > T_u. A rule of thumb that justifies the use of a single check is that torsional resistance rarely governs beam design, and checking provision 11.6.1 of ACI 318-83 usually shows the base torsional resistance of the beam is adequate to resist torsion.

The ingredients of the test are the factored applied torque on the member, T_u, and the member base torsional strength. Satisfaction of the test requires that the applied torque be less than the strength reduced base torsional capacity of the beam. T_u is obtained from the user. The base torsional strength is computed using the following equation (subsection 11.6.1 of ACI 318-83).

$$BaseTorsionalResistance = \Phi(0.5\sqrt{f'_c}\Sigma x^2 y)$$

Shape	Criteria					
	Name	Problem Type	Supposed Greater Ingredient	Greater Method	Supposed Lesser Ingredient	Lesser Method
T, L, Rect-angle	base torsion sufficient	rein. concrete	member base torsion	compute	Tu	not com

Table 6-7: Relevant Criteria for RC Beams in Torsion

6.3. Summary

Enhancement of the system to handle the reinforced concrete beam problem type, required addition of approximately thirty rules that identify the relevant parameters and evaluation criteria, obtain specific parameter values, and compute criteria ingredients for flexural, shear, and torsional evaluation of reinforced concrete rectangular, T and L sections.

Chapter Seven
Summary and Discussion

SEES represents a large problem, and to make it a realistic application in industry, the extent of the system's limitations must be considered. This chapter summarizes SEES and takes a critical look at it. The first section summarizes the computer implementation and expertise in the current prototype. The next section reports some of the major assumptions and limitations of the system. The third section gives evaluation and criticisms of the program. The final section lists possible enhancements to the system.

7.1. Summary of Project

The purpose of this project is to design and implement a tool for the evaluation of existing structures using expert system techniques. The design involves compiling engineering experience and knowledge into a general problem solution process. The experience and knowledge needed is engineering heuristics on evaluation practice, knowledge of past and current design codes, and knowledge of handbook solution formats. The implementation involves formalizing the process and knowledge in a rule-based expert system called SEES (Strength Evaluation of Existing Structures).

7.1.1. Summary of Computer Implementation

SEES was implemented using a goal driven control strategy in a GC-Lisp version of OPS5 [Forgy 81]. A generic problem solving rule base exists that is readily expandable to a large set of problem types. Expansion involves including heuristics for each specific problem type. The current prototype evaluates existing reinforced concrete beam members.

The evaluation problem is decomposed into three major goals: formulation, redesign, and evaluation. The first goal, formulation, gets the problem specification from the user. For the redesign goal, the problem specification information is used to determine the list of relevant design parameters and request or infer parameter values. For evaluation, the relevant criteria are identified, their ingredients identified and obtained, and each criteria checked for satisfaction.

7.1.2. Summary of Expert Knowledge Acquisition

Knowledge used in SEES is a combination of problem solving knowledge, code requirement knowledge, and expert heuristic knowledge. Problem solving knowledge was obtained partly through collaboration with experts and through texts on reinforced concrete design. Problem solving knowledge was used primarily for generating the general problem solving control structure of the system. Code requirements knowledge was obtained from the American Concrete Institute Building Regulations for Reinforced Concrete committee 318, 1983. This code knowledge is used during the evaluation portion of the program. Expert heuristic knowledge was obtained through research of old code specifications and question and answer sessions with professional engineers. Expert knowledge played a major role in the redesign portion of the program where design parameter values are obtained.

7.2. Assumptions and Limitations

Although the program works and produces results, it is not flawless. Some assumptions and limitations had to be imposed on the system in order to produce the prototype. This section describes the assumptions and limitations.

7.2.1. Use of Codes

SEES evaluates member capacity according to current code criteria using parameter values obtained from practices documented in past codes. The underlying assumption is that the design engineer who performed the original design, did so in complete accordance to the current code at the time. This assumption is extended to the assumption that the engineer used the ACI or AISC codes to perform the design. The questions that arise are:

1. Did the engineer adhere to the established codes?
2. Did the engineer use his own judgment?
3. Did the engineer follow the philosophy of a different design handbook?

For the time being, it is safe to assume the original design engineers adhered to the codes and be conservative wherever else it is possible.

7.2.2. Incomplete Expertise

The expertise represented in the present prototype of SEES is not as complete as it should be. This is partly due to the fact that experts were not available to spend a large amount of time providing example problems and heuristics to help with the details of the problem solution. For this reason, the basic problem solution behavior of SEES was derived from text book examples and the interpretation of the author. Most of the expertise incorporated into the system was obtained through quick question and answer sessions with experienced professional structural engineers about individual problem details, not entire problem solving paradigms. This difficulty in obtaining expert knowledge is typical of the current state of knowledge acquisition by knowledge engineers. It will not change until experts have the time and motivation to devote themselves more fully.

7.3. Evaluation and Criticism

The following section discusses some of the aspects of SEES that should be changed.

7.3.1. Solution Paradigm

As engineering becomes more oriented toward use of computer tools, engineers are faced with difficult decisions on how to model problem approaches in computer code that result in efficient but clear use of computing power. In expert system development, two problem solving strategies are the data driven and goal driven strategies. In a goal driven strategy, flow of control is based on the satisfaction of an ordered or unordered set of goals. The intention behind data driven strategies is that control is based on reaction to a predefined input data set.

Typically, the task oriented nature of engineering problems suggests the use of goal driven problem approaches. The idea here is that engineers prefer not to have all data input at the beginning of a system run, but would rather acquire pieces of data only as needed to satisfy goals. Even experts do not know all of the pieces of data needed to solve the problem at problem outset.

The top level of SEES resembles a data driven approach since parameter identification occurs before criteria identification. A result of this is that the programmer must take special care in coding to assure that all possible parameters are identified in the redesign goal to prevent the system getting caught later in the evaluate goal when it must use parameter values for computation of an ingredient. To make SEES behave like the pure goal driven nature of engineering experts, an alternate approach to the system goal structure in Figure 5-2 is shown in Figure 7-1 where identification of relevant design parameters occurs during evaluation. Essentially, the entire redesign branch of the goal tree has been relocated as a subgoal of the

identify_ingredients goal in the evaluation goal. Here, the parameters are not identified and obtained until the system is sure that the parameter is needed for an ingredient of an evaluation test criteria. This revised goal tree provides efficiency to SEES, making sure that minimum knowledge is utilized.

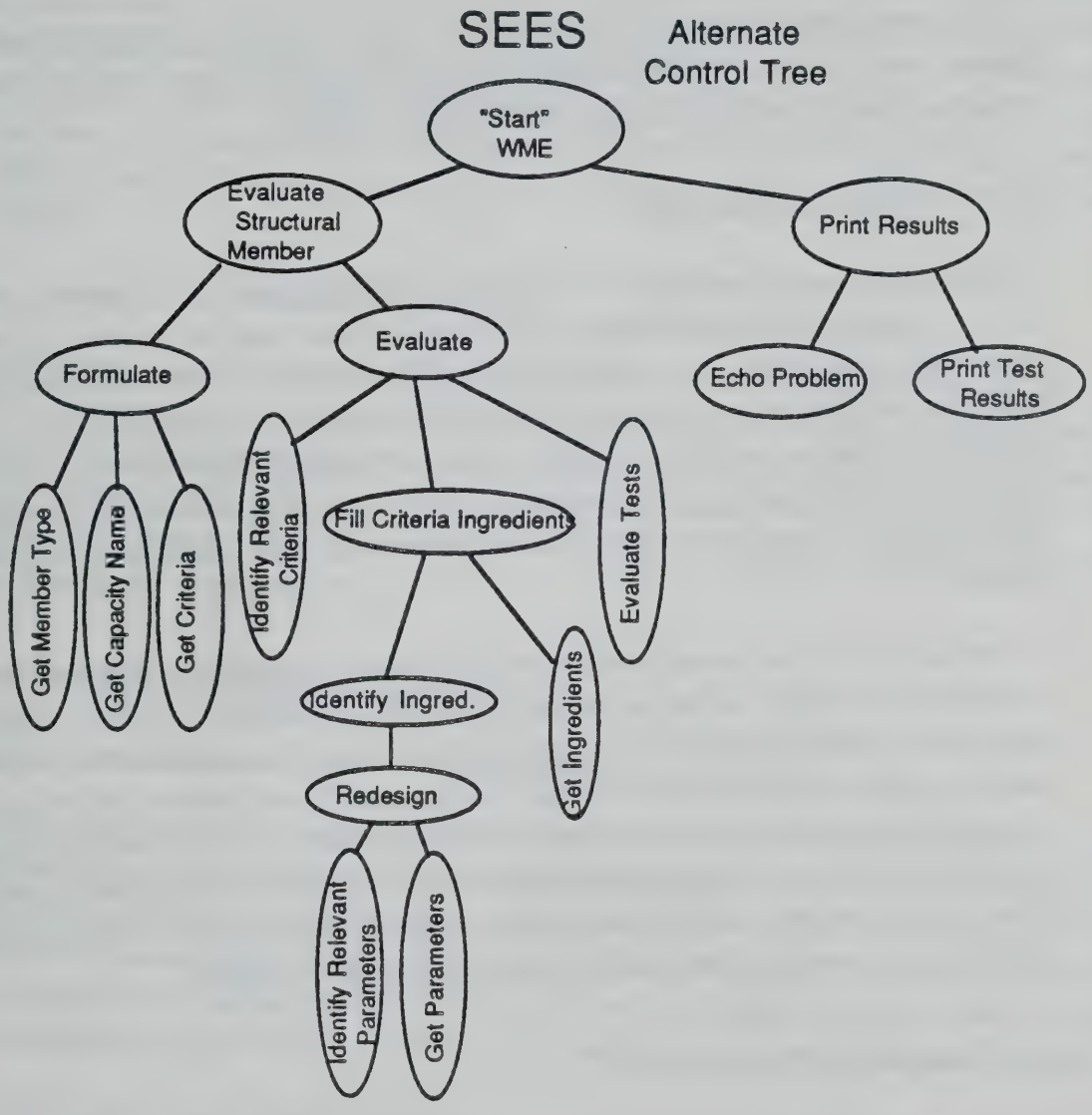

Figure 7-1: Modified SEES Control Tree to Propose Pure Backward Chaining

---

The decision of which is the best way to program the SEES control structure should be based on efficiency and clarity. Satisfaction of both of these criteria is difficult in OPS5, an inherent data-driven language. Two questions remain:

1. Should the problem solution be modeled after the control strategy provided in the production system language?
2. Should the clarity of the production rules be sacrificed to implement the most appropriate solution strategy?

The current prototype does both of the above. The top level control is based on a general goal driven strategy. A clearer implementation would use a control strategy closer to Figure 7-1 and would sacrifice efficiency by making some rules more specific.

7.3.2. Programming Methods

SEES is in its second prototype; it is not polished. Programming in OPS5 offers a multitude of ways to perform similar programming tasks and the best way is not always chosen first. Programmers go through a process of iterative programming in OPS5 where productions are evaluated and new ways found to do the job more efficiently. The first sentence of the last paragraph in Chapter 4 of [Brownston et al 86] states: "Large production systems such as expert systems typically grow in waves, incremental growth alternating with sudden shrinkage." As SEES grows to handle more of its intended capabilities, it will undoubtedly go through this process.

There are parts of the present prototype which deserve reconsideration, particularly the portions of code that perform the list processing of identified parameters and test criteria. It is questionable whether there is need to have a set of specific rules returning unique lists of identified data items that correspond to a particular problem definition and then iteratively decompose those lists to make the goals for obtaining values. A simpler methodology would be to have the same type of rules that returned the lists create the working memory elements for those parameters and have the goal setting rule merely check for their instances in working memory. An advantage to the list processing rules is the clarity of code.

7.4. Possible Enhancements

This section proposes enhancements to SEES to make it a better system. These enhancements are considered in two groups: additions that aid implementation efficiency and additions that enhance evaluation capability.

7.4.1. Implementation Efficiencies

7.4.1.1. Decision Tables

One of the primary goals in further development of SEES should be to have a definite separation of the system process knowledge from the heuristic knowledge by using a decision table format for heuristics. This can be done by developing a tree of decision tables where the top level decision table returns the single value of the members integrity and all lower level decision tables concentrate on obtaining the values and computing the intermediate results to aid that decision. OPS5 provides a decision table evaluator, DTE, capable of evaluating decision table trees. DTE was written by Jim Garret in the Department of Civil Engineering at Carnegie Mellon University. DTE was not translated from Franzlisp into GC-Lisp, and therefore is not used in SEES.

7.4.1.2. Consistency Checking of User Input

At this point in time, there is little consistency checking of user input that insures compatibility of data. For example, there are no constraint handling rules that check to see if user specifies bar sizes that satisfy spatial limits of the member. Further, there are no rules that check the user for unrealistic data values. If the user chose to, he could input a steel strength of 1, and SEES would still generate results. Checking the validity of user input is important to both the accuracy and user friendliness of the program.

7.4.2. Additional Evaluation Capabilities

7.4.2.1. Consideration of Special Condition Criteria

SEES should be able to evaluate structural integrity based on the physical condition of the member due to aging of the constituent materials and weathering. Old structural members are rarely in the condition they were just after construction. For this reason, the accuracy of SEES requires that results take into account the aging of members by factoring or some other means. The present prototype does not perform this type of result modification since the available experts for consultation did not have the confidence to recommend a mechanism for such modifications.

7.4.2.2. Member Strengthening Suggestions

SEES should suggest how to strengthen a member with an unsatisfied evaluation. This would require a large amount of extra domain knowledge and query about the intended future use of the member, but would provide valuable insight that is otherwise not available to inexperienced engineers.

7.4.2.3. Certainty Factors

Incorporation of certainty factors in expert systems is popular because it gives the user a level of confidence on which he can rely on results. There are two problems with certainty which provide enough reason to exclude such factors from the current prototype of SEES.

1. Existing techniques do not guarantee accurate or appropriate results after certainty propagation through multiple levels of expert inference.
2. Interpretation of the level of belief is subjective, defined by the system programmer according to his own scale and the user may interpret the scale in a completely different manner.

Until more is known about uncertain inference techniques, use of certainty factors in SEES may only distort the users interpretation of results.

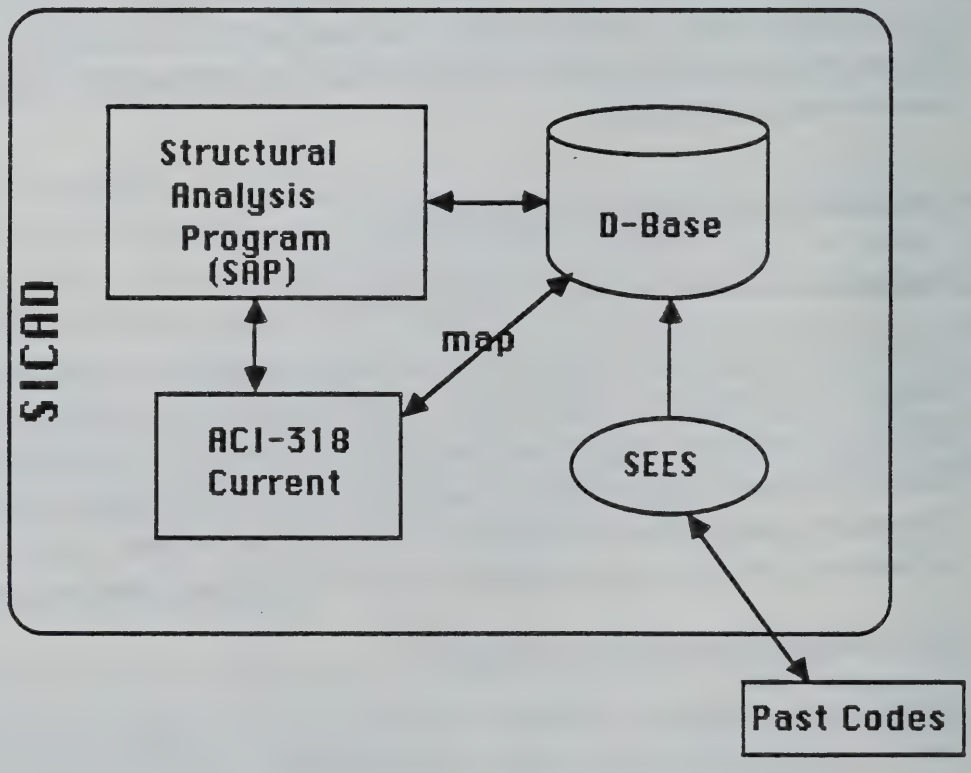

Figure 7-2: Possible Use of SEES Module on Larger Analysis Program as per discussion with K. Reed, NBS

7.4.2.4. SEES as a Front End

Whenever working stress method is used in design, allowable stresses have to be greater than actual stresses to satisfy design criteria. In order to calculate actual stresses, applied loads must be known as well as section properties of the member. Determining the applied loads may be difficult though if the member is part of a structural system where load re-distribution is a function of individual member stiffnesses. It is difficult for SEES to compute accurate member stresses directly from the inferred set of member properties. A more feasible solution would be for SEES to input the inferred section properties to another program that could perform the structural analysis of the entire system of members taking into account the section properties of all members for moment re-distribution purposes, and then check the member adequacy. In this case, SEES becomes a module of a larger system along with a structural analysis and design program. Figure 7-2 shows how SEES might fit into a large program like SICAD [Lopez,Elam 87]. SEES would provide the design parameters for members of existing structural systems, then SICAD would perform the structural analysis and evaluation of the members.

Appendix A
Selected OPS5 Rules

Provided in this appendix are portions of the system rules that highlight the implementation. An important process rule is provided that was not covered in Chapter 5. The general rules for parameter acquisition are given. An example of specific parameter acquisition rules are provided. The criteria test checking rules are provided. Examples specific problem solving rules for reinforced concrete beams are given. Finally, the file containing the system's permanent domain working memory knowledge is provided. The remainder of the SEES code is available through Professor Mary Lou Maher at Carnegie Mellon University.

A.1. Control Rules

Most control rules have already been illustrated in Chapter 5. Below is the rule that decomposes the list of relevant criteria during evaluation.

```
(p make_the_fill_ingred_and_eval_goals_from_criteria_list
;*****************************************************************************
;  Abstract:  The following rule uses an iterative process to make the
;  fill_ingredient_value and criteria evaluation goals for the list of
;  criteria that are relevant to the particular problem.
;
;  Rule:
;  IF there is an active goal to identify the relevant criteria
;  AND there are relevant criteria names
;  AND there are criteria_problem_relations
;  AND there is a list of supposed greater ingredient names
;  AND there are supposed_greater_methods
;  AND there is a list of supposed lesser ingredient names
;  AND there are supposed lesser methods
;  AND the rel_criteria_index has a non nil index
;  AND the index is not zero yet
```

```
;  THEN make the goal evaluate_criteria with task equal to the substructure
;       (substr) of the list representing the item at the current index
;       position of the criteria list created in the identification goal.
;       And fill in the corresponding supposed lesser and greater, and
;       their corresponding methods.  And fill in the criteria problem
;       relations.
;       Make the goal with status active.
;  AND make the goal to fill the eval ingredients under the same task with
;        status active
;  AND modify the index number to the current one minus one.
;*****************************************************************************
(goal ^name jump_to_evaluating_criteria ^status active)
{(relevant_criteria_names)<relevant_criteria>}
{(criteria_problem_relations)<criteria_relations>}
{(supposed_greater_crit_ingredients)<greaters>}
{(supposed_greater_acquire_methods)<greater_methods>}
{(supposed_lesser_crit_ingredients)<lessers>}
{(supposed_lesser_acquire_methods)<lesser_methods>}
{(rel_criteria_index ^index { <curr_index> <> nil } )<rel_criteria_index>}
(rel_criteria_index ^index { <curr_index> <> 0 } )
-->
(bind <item> ( compute ( <curr_index> + 1 )))
(bind <criteria_name> (substr <relevant_criteria> <item> <item>))
(bind <problem_relation> (substr <criteria_relations> <item> <item>))
(bind <greater_ingredient> (substr <greaters> <item> <item>))
(bind <greater_method> (substr <greater_methods> <item> <item>))
(bind <lesser_ingredient> (substr <lessers> <item> <item>))
(bind <lesser_method> (substr <lesser_methods> <item> <item>))
(make goal ^name evaluate_criteria
           ^task <criteria_name>
           ^relative_to <problem_relation>
           ^status active)
(make goal ^name fill_eval_criteria
           ^task <criteria_name>
           ^relative_to <problem_relation>
           ^status active)
(make eval_criteria ^name <criteria_name>
                    ^relative_to <problem_relation>
                    ^supposed_greater <greater_ingredient>
                    ^greater_method <greater_method>
                    ^supposed_lesser <lesser_ingredient>
                    ^lesser_method <lesser_method>
                    ^status empty)
(bind <next> (compute ( <curr_index> - 1 )))
(modify <rel_criteria_index> ^index <next>)
)
```

A.2. General Get Parameter Rules

When parameter acquisition follows the general approach discussed in Chapter 5, the following ten rules apply.

```
;=============================================================================
;   The following rules are general rules devoted to the
;  acquisition of design parameters.  The general rules under this
;  section are
;    1)for the user knowledge query of one of the parameters.
;    2)for the asking of the value of the parameter if the user knows it.
;    3)the setting of the sub goal to infer the parameter if it can be inferred
;      and the user does not know it.
;    4)checking the user satisfaction of an inferred parameter.
;    5)re-firing fill if the user is unsatisfied with an inferred parameter.
;    6)making the goal to get the the year of construction if it is needed
;      and is not known yet.
;    7)warning the user that the system cannot proceed unless the value of
;      a non-inferred parameter can be input.
;    8)refiring fill if the user wishes to proceed
;    9)set the goal to compute a parameter for those parameters that are
;      meant to be computed and not queried for.
;   10)set the goal status for get_parameter to complete for those parameters
;      that are computed, once they are computed.
;=============================================================================

(p check_user_knowledge_of_parameter
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule checks to see if the user knows the value of one of the
 ;  parameters which is the task of one of the goals to get parameters.
 ;    There are some parameters though that through the method attribute field
 ;  of parameter_information, force automatic computing and avoid user query.
 ;
 ;  Rule:
 ;  IF the goal is to get parameters with task of <name>
 ;  AND parameter_information with <name> shows a method of not compute
 ;        -all others we want to be asked for
 ;  AND there is a parameter_query named <name> that is unknown
 ;  THEN ask the user if he knows it
 ;*****************************************************************************
(goal ^name get_parameters ^task <name> ^status active)
(parameter_information ^name <name> ^method { <> compute })
{(parameter_query ^name <name> ^status empty)<parameter_query>}
-->
(owrite (crlf)(crlf)See if user knows the value of <name>)
(ocall fill (substr <parameter_query> 1 inf))
(oremove <parameter_query>)
)

(p ask_for_value_of_parameter
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule asks the user for the value of a parameter if the user
 ;  actually knows it.
 ;
 ;  Rule:
 ;  If the goal is to get parameters with task <name>
 ;  AND there is parameter_information named <name> with method i
 ;  AND the user knows what the parameter is
 ;  THEN ask the user for the value and store it
 ;*****************************************************************************
{(goal ^name get_parameters ^task <name> ^status active)<goal>}
(parameter_information ^name <name> ^method i)
(parameter_query ^name <name> ^user_knows yes ^status filled)
{(parameter_value ^name <name> ^status empty)<parameter_value>}
-->
```

```
(ocall fill (substr <parameter_value> 1 inf))
(oremove <parameter_value>)
(modify <goal> ^status complete)
)

(p give_user_option_if_ignorant_about_non_inferring_parameter
 ;*****************************************************************************
 ;  Abstract:  This rule checks to see if the user answered no to the question
 ;  of whether he knows the value of a parameter that cannot be inferred.
 ;  If so, the system reports that it cannot do anything without the user
 ;  knowledge of this basic parameter and then calls Fill to ask the user if
 ;  he wishes to continue or not. If the user wishes to continue then the goal
 ;  is modified to active again.
 ;
 ;  Rule:
 ;  IF there is an active goal to get parameters with task <name>
 ;  AND there is parameter query information showing user knowledge of no
 ;  AND there is parameter information showing that <name> cannot be inferred
 ;  AND there is continuance working memory class
 ;  THEN write that the system needs this info from the user in order to
 ;       continue
 ;  AND call fill to see if the user wishes to continue
 ;*****************************************************************************
(goal ^name get_parameters ^task <name> ^status active)
(parameter_query ^name <name> ^user_knows no)
(parameter_information ^name <name> ^inferable no)
{(continuance ^status empty)<continuance>}
-->
(owrite(crlf)(crlf)It is imperative that the user be able to supply the)
(owrite(crlf)value of <name> which is not inferable)
(owrite(crlf)*Otherwise it cannot complete its objective)
(owrite(crlf)*You will have to supply this information for the system to run)
(ocall fill (substr <continuance> 1 inf))
(oremove <continuance>)
)

(p restart_parameter_acquisition_upon_users_wish
 ;*****************************************************************************
 ;  Abstract:  This rule re-queries the user for a design parameter if the
 ;  user originally said that he did not know it but then said he wished
 ;  to continue.
 ;
 ;  Rule:
 ;  If there is an active goal to get parameters with task <name>
 ;  AND there is a parameter query named <name> with user knows equal to no
 ;  AND there is a parameter information with name <name> and inferable no
 ;  AND there is parameter value with <name> and status empty
 ;  AND there is continuance with value of yes
 ;  THEN modify the parameter query to user_knows yes
 ;  AND modify the continuance back to empty for the next parameter
 ;*****************************************************************************
(goal ^name get_parameters  ^task <name> ^status active)
{(parameter_query ^name <name> ^user_knows no)<parameter_query>}
(parameter_information ^name <name> ^inferable no)
(parameter_value ^status empty)
{(continuance ^value yes ^status filled)<continuance>}
-->
(owrite (crlf)(crlf)<name> will be re-asked for)
(modify <parameter_query> ^user_knows yes)
(oremove <continuance>)              ;with an assigned value of value
(make continuance ^status empty)     ;for next parameter in this situation
)
```

```
(p going_to_have_to_infer_parameter
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule sets a new goal to infer the parameter if it can be
 ;  inferred and the user does not know it and it is not known.
 ;
 ;  Rule:
 ;  IF goal is to get parameters with task <name> and relative to <relation>
 ;  AND <name> is inferable
 ;  AND <name> is not known by the user
 ;  AND <name> is not known
 ;  THEN make the goal to infer_parameter active with a task <name> and
 ;         relative_to <relation>
 ; *****************************************************************************
(goal ^name get_parameters ^task <name> ^relative_to <relation>
      ^status active)
(parameter_query ^name <name> ^user_knows no)
(parameter_information ^name <name> ^inferable yes ^method i)
(parameter_value ^name <name> ^status empty)
-->
(make goal ^name infer_parameters ^task <name> ^relative_to <relation>
           ^status active)
(owrite (crlf)(crlf) <name> will have to be inferred)
)


(p check_user_satisfaction_of_inferred_parameter
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule asks the user whether he is satisfied with the value
 ;  of an inferred parameter.  If the user is not, this rule sets up the
 ;  immediate goal to prompt the user for the override value.
 ;
 ;  Rule:
 ;  IF the goal is to infer parameters with task <name> and status complete
 ;  AND there is parameter info with same <name> that has been inferred
 ;  AND that parameter has a value
 ;  AND the satisfaction of the parameter value is nil
 ;  THEN ask the user if he is satisfied with the value
 ;  AND accept his response into the satisfied attribute of parameter info
 ;*****************************************************************************
(goal ^name infer_parameters ^task <name> ^status complete)
{(parameter_information ^name <name> ^inferred yes ^over_ride nil)
     <parameter_info>}
(parameter_value ^name <name> ^value <value>)
-->
(owrite (crlf)The parameter <name> has been inferred)
(owrite (crlf)to a value of <value>)
(owrite (crlf)(crlf)Do you wish to over-ride this value with one of your own?)
(owrite (crlf)y or n(crlf))
(bind <over_ride> (accept))
(modify <parameter_info> ^over_ride <over_ride>)
)

(p refire_fill_if_user_unsatisfied
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule modifies the status of the get parameter goal to active
 ;  again and resets the values of that parameters attributes to nil so the
 ;  ask_for_value_of_parameter rule will fire again for that goal.  This
 ;  rule is fired if the user is unsatisfied with the inferred value of
 ;  a parameter.
```

```
 ;
 ;  Rule:
 ;  IF goal to get parameters with task <name> is complete
 ;  AND parameter info with <name> has a value of "y" for over_ride
 ;  AND parameter value with that <name> has a value
 ;  -->
 ;  THEN modify the goal to get parameters active
 ;  AND modify the value to nil and status to empty
 ;*****************************************************************************
{(goal ^name infer_parameters ^task <name> ^status complete)<goal>}
(parameter_information ^name <name> ^over_ride y)
{(parameter_query ^name <name>)<parameter_query>}
{(parameter_value ^name <name> ^value <value> ^status filled)
    <parameter_value>}
-->
(modify <goal> ^name get_parameters ^status active)
(modify <parameter_query> ^user_knows yes)
(modify <parameter_value> ^value nil ^status empty)
)


(p make_goal_to_get_year_if_it_is_not_known_yet
 ;*****************************************************************************
 ;  Abstract:   This rule makes the goal to get the year of construction
 ;  if it is not known yet and a parameter  must be inferred by the
 ;  year of construction.
 ;
 ;  Rule:
 ;  IF the goal is to infer parameters with task <name>
 ;  AND the criteria for name is year
 ;  AND name is not known yet
 ;  AND the year is not known yet
 ;  THEN make the goal to get the year of construction
 ;  AND write that the year must be gotten for inference of <name>
 ;*****************************************************************************
(goal ^name infer_parameters ^task <name> ^status active)
(parameter_information ^name <name> ^criteria year)
(parameter_value ^name <name> ^status empty)
(year_query ^status empty)
(year_built ^status empty)
-->
(owrite (crlf)(crlf)The year of construction must be known for the)
(owrite (crlf)estimation of <name>)
(make goal ^name get_year  ^status active)
)

(p set_goal_to_compute_a_compute_parameter
 ;*****************************************************************************
 ;  Abstract:  This rule sets the goal to compute parameters for those
 ;  parameters that were specified for computing in the method attribute of
 ;  their parameter_information wme's.  These are the parameters for
 ;  which it is impractical to go thru the usual steps of user query and
 ;  inference.
 ;
 ;  Rule:
 ;  IF there is a goal to get parameters with task <name> thats active
 ;  AND there is parameter information named <name> with method = compute
 ;  THEN make the goal to compute_parameters with task <name>, status active
 ;  AND write that the parameter <name> will be computed
 ;*****************************************************************************
(goal ^name get_parameters ^task <name> ^status active)
(parameter_information ^name <name> ^method compute)
```

```
-->
(make goal ^name compute_parameters ^task <name> ^status active)
(owrite (crlf)(crlf)The parameter named - <name> - will be computed directly)
(owrite (crlf)instead of following normal routine of user query)
(owrite (crlf)**Computation will occur as soon as the constituent data items
             become available)
)

(p complete_a_get_parameter_goal_from_compute_goal
 ;*****************************************************************************
 ;  Abstract:  This rule is a goal cleanup rule.  It fires when a computed
 ;  parameter, originally under a get_param goal, has its compute_parameter
 ;  goal completed.  It then completes its get_parameter goal.  The purpose
 ;  of this rule is hopefully make the system run faster later on for those
 ;  rules that check for get_parameter goals in their first premises.
 ;
 ;  Rule:
 ;  If there is a goal to get_parameters with task <name> thats active
 ;  AND there is a goal to compute_parameters with task <name> thats complete
 ;  THEN make the get_parameter goal complete
 ;*****************************************************************************
{(goal ^name get_parameters ^task <name> ^status active)<goal>}
(goal ^name compute_parameters ^task <name> ^status complete)
-->
(modify <goal> ^status complete)
)
```

A.3. Example of Specific Parameter Acquisition Rules

The following 8 rules demonstrate the acquisition of the design parameter, area of longitudinal tension steel for reinforced concrete.

```
(p set_goal_to_get_rebar_specs_if_getting_d_or_As
 ;*****************************************************************************
 ;  Abstract:  This rule sets the goal to get the rebar specs if there is a
 ;  goal to compute_parameters with task of As or d and the get rebar specs
 ;  goal has not been set yet.
 ;
 ;  Rule:
 ;  IF there is and active goal to compute_parameter with task of d or As
 ;  AND there is no goal to get rebar specs yet
 ;  THEN initialize the rebar_info wme to empty
 ;  AND make the goal to get the rebar specs
 ;*****************************************************************************
(goal ^name compute_parameters ^task << As d >> ^status active)
-(goal ^name get_rebar_specs)
-->
(make rebar_info ^diameter_status empty ^status empty)
(make goal ^name get_rebar_specs ^status active)
(owrite(crlf)(crlf)Specs on the longitudinal reinforcing steel will now be
                   pursued)
)


(p get_rebar_quantity
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule sets the goal to get the number of bars if the goal
 ;  is to get the rebar and there is a goal to get As and the rebar desig.
 ;  is known.
 ;
 ;  Rule:
 ;  IF the goal is to get the rebar
 ;  AND the goal is to compute_parameters with task As
 ;  AND there is a parameter_query named rebar number that the user knows
 ;  THEN make the working memory elements for rebar number
 ;  AND make the subgoal to get the rebar specs.
 ;*****************************************************************************
(goal ^name get_rebar_specs ^status active)
(goal ^name compute_parameters ^task As ^status active)
(parameter_query ^name rebar_number ^user_knows yes ^status filled)
(parameter_query ^name rebar_shape ^user_knows yes ^status filled)
-->
(make parameter_information ^name number_of_bars  ^inferable no ^method i)
(make parameter_query ^name number_of_bars ^status empty)
(make parameter_value ^name number_of_bars ^status empty)
(make goal ^name get_parameters ^task number_of_bars ^status active)
)

(p compute_As
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule computes the value of As if the goal is to get As and the
 ;  rebar query was successful and the values of bar area and quantity are
 ;  known and the area of steel is not known yet.
 ;
 ;  Rule:
 ;  IF the goal is to compute_parameters with task As
 ;  AND the area of steel is not known yet
 ;  AND the rebar info has been filled
 ;  AND the parameter value named number of bars has a value
 ;  THEN compute As by #bars * bar area
 ;  AND modify the parameter value named As to the value
 ;  AND close the goal
```

```
;*****************************************************************************
{(goal ^name compute_parameters ^task As ^status active)<goal>}
{(parameter_value ^name As ^value nil ^status empty)<parameter_value>}
(rebar_info ^status filled ^bar_area <bar_a>)
(parameter_value ^name number_of_bars ^status filled ^value <num_bars>)
-->
(bind <area_steel> (compute (<bar_a> * <num_bars>)))
(owrite (crlf)(crlf)The area of steel computed for <num_bars> bars)
(owrite (crlf)with bar area of <bar_a> square inches is)
(owrite (crlf)equal to <area_steel> square inches)
(modify <parameter_value> ^value <area_steel> ^status filled)
(modify <goal> ^status complete)
)


(p modify_to_ask_As_if_rebar_specs_unavail
;*****************************************************************************
;  Abstract:
;    This rule modifies the wme's of As in order to get the area of steel
;  from the user by asking directly, if it turns out that the user does not
;  know either the parameter values of rebar_number or number_of_bars or
;  rebar_shape.  It
;  checks using an "exclusive-or" of the user knowledge
;  on three possible rebar inputs.  If
;  the user does not know any one of the three, the system assumes the
;  user knows nothing of the rebar specs.
;
;  Rule:
;  IF the goal is to compute_parameters with task As
;  AND the parameter_query named As is empty still
;  AND the user does not know the bar number or number of bars or shape
;  THEN keep the parameter_query the same
;  AND modify the parameter_information to method i
;  AND write a message
;  AND modify the goal to failed
;*****************************************************************************
{(goal ^name compute_parameters ^task As ^status active)<goal>}
{(parameter_query ^name As ^status empty)<parameter_query>}
(parameter_query ^name << rebar_number rebar_shape number_of_bars >>
                 ^user_knows no ^status filled)
{(parameter_information ^name As ^method <method>)<parameter_information>}
-->
(owrite (crlf)(crlf)Assume user does not know rebar specification so)
(owrite (crlf)ask the area of steel directly)
(modify <parameter_information> ^method i)
(modify <goal> ^status failed)
)


(p infer_area_of_steel_to_be_one_percent
;*****************************************************************************
;  Abstract:  This rule performs the last acquisition of As after all previous
;  attempts fail.  If the user did not know the bar specs, and was not able
;  to answer to the steel area directly, then this rule computes the value of
;  the steel are by assuming a one percent reinforcement ratio.  This is
;  a reasonable guess.
;
;  Rule:
;  IF there is an active goal to infer_parameters with task As
;  AND As is not known yet
;  AND there is parameter_information on the parameter
;  AND the member width b or bw is known
```

```
 ;  AND the member depth d is known
 ;  THEN compute the area of steel as being one percent of the member area
 ;  AND give the result to As
 ;  AND write what you just did
 ;  AND modify the parameter_information to inferred - yes
 ;  AND complete the goal
 ;*****************************************************************************
{(goal ^name infer_parameters ^task As ^status active)<goal>}
{(parameter_value ^name As ^value nil ^status empty)<parameter_value>}
{(parameter_information ^name As)<parameter_information>}   ;for binding
(parameter_value ^name << b bw >> ^value <b> ^status filled)
(parameter_value ^name d ^value <d> ^status filled)
-->
(bind <result> (compute 0.01 * <b> * <d> ))
(modify <parameter_value> ^value <result> ^status filled)
(owrite (crlf)(crlf)User unable to specify area of steel information)
(owrite (crlf)**System will assume existence of at least one percent steel in
                member)
(owrite (crlf)Area of tension steel used computed from web width of <b> inches
              and member)
(owrite (crlf)depth of <d> inches)
(owrite (crlf)**One percent yields a steel area of <result> square inches)
(modify <parameter_information> ^inferred yes)
(modify <goal> ^status complete)
)


(p get_rebar_shape_and_number_for_long_steel
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule sets the subgoals to get the rebar specifications if the goal
 ;  is to get the rebar and the rebar specs have not been queried yet.
 ;    The basic rebar specs are the bar number and shape.
 ;
 ;  Rule:
 ;  IF the goal is to get the rebar specs
 ;  AND the rebar number has not been asked
 ;  AND the rebar shape has not been asked yet
 ;  AND the rebar info is not known yet
 ;  THEN make the wme for the rebar number
 ;  AND make the subgoals to get the rebar shape and number
 ;*****************************************************************************
(goal ^name get_rebar_specs ^status active)
(rebar_info ^bar_d nil ^bar_shape nil ^diameter_status empty)
-->
(make parameter_information ^name rebar_number ^inferable i ^method i)
(make parameter_query ^name rebar_number ^status empty)
(make parameter_value ^name rebar_number ^status empty)
(make parameter_information ^name rebar_shape ^inferable i
                            ^method ask_separate)
(make parameter_query ^name rebar_shape ^status empty)
(make rebar_shape_info ^status empty)
(make goal ^name get_parameters ^task rebar_shape ^status active)
(make goal ^name get_parameters ^task rebar_number ^status active)
(owrite (crlf)(crlf)User must be asked the rebar number if possible)
(owrite (crlf)User must also be asked if shape of rebars is round or square)
)

(p ask_for_shape_of_long_rebar
 ;*****************************************************************************
 ;  Abstract:
 ;    This rule calls fill for the rebar shape.
```

```
;
;  Rule:
;  If the goal is to get parameters with task rebar_shape
;  AND there is parameter_information named shape with method ask_separate
;  AND the user knows what the shape of the rebar is
;  THEN call fill to get the rebar shape
;*****************************************************************************
{(goal ^name get_parameters ^task rebar_shape ^status active)<goal>}
(parameter_information ^name rebar_shape ^method ask_separate)
(parameter_query ^name rebar_shape ^user_knows yes ^status filled)
{(rebar_shape_info ^shape nil ^status empty)<rebar_shape_info>}
-->
(ocall fill (substr <rebar_shape_info> 1 inf))
(oremove <rebar_shape_info>)
(modify <goal> ^status complete)
)

(p set_dia_and_area_of_bar_for_long_steel
;*****************************************************************************
;  Abstract:
;    This rule fills in the value of the rebar info working memory element
;  if the value of the bar number and shape is known.
;
;  RULE:
;  IF goal named get parameters with task rebar_number
;  is complete
;  AND the same is true for the get_parameters goal with task rebar shape
;  AND there is a parameter value named rebar_number with a value
;  AND there is rebar_shape_info with a value
;  AND there is a wme in the rebar_specs with that rebar_number and shape
;  THEN modify the rebar_info attribute number to the rebar_number
;  AND modify the rebar_info attribute shape to the rebar_shape
;  AND modify the rest of the rebar_info attributes to the
;    corresponding values in the applicable wme in the rebar database
;*****************************************************************************
(goal ^name get_parameters ^task rebar_number ^status complete)
(goal ^name get_parameters ^task rebar_shape ^status complete)
(parameter_value ^name rebar_number ^value <num> ^status filled)
(rebar_shape_info ^shape <shape> ^status filled)
(rebar_specs ^shape <shape> ^number <num> ^diameter <diameter> ^area <area>)
{(rebar_info ^bar_shape nil ^bar_num nil ^bar_d nil ^diameter_status empty
             ^bar_area nil  ^status empty)<rebar_info>}
-->
(owrite (crlf)(crlf)For the <shape> number <num> bar a diameter of <diameter>)
(owrite (crlf)inches and bar area of <area> square inches is used)
(modify <rebar_info> ^bar_shape <shape> ^bar_num <num> ^bar_d <diameter>
                     ^bar_area <area> ^diameter_status filled ^status filled)
)
```

A.4. Criteria Test Checking

The following two general rules perform the boolean tests for the relevant criteria. The first rule checks for satisfaction of the test and the second rule checks for dissatisfaction of the test.

```
(p check_satisfaction_of_criteria_test
 ;*****************************************************************************
 ;  Abstract:  This rule is a general rule that checks for the satisfaction
 ;  of an identified evaluation criteria.
 ;    This rule checks to see if the
 ;  supposed greater of the two ingredients is in fact the greater of the two.
 ;  If so, it assigns a value of yes to the satisfied attribute of the eval
 ;  criteria.
 ;
 ;  Rule:
 ;  IF there is an active goal to evaluate_criteria with task <criteria_name>
 ;  AND there is an eval_criteria wme with name <criteria_name> and a
 ;   supposed_greater of <greater> and supposed_lesser of <lesser> thats empty
 ;  AND there is a crit_ingredient wme wit name <greater> thats filled
 ;  AND there is a crit_ingredient wme with name <lesser> thats filled
 ;  AND <greater> is greater than or equal to <lesser>
 ;  THEN modify the eval_criteria to status filled and give its result field
 ;        a value of satisfied
 ;  AND modify the goal to complete
 ;  AND tell of the results.
 ;*****************************************************************************
{(goal ^name evaluate_criteria ^task <criteria_name> ^status active)<goal>}
{(eval_criteria ^name <criteria_name> ^supposed_greater <ingred_one>
                                      ^supposed_lesser <ingred_two>
                                      ^result nil
                                      ^status empty)<eval_criteria>}
(crit_ingredient ^name <ingred_two> ^value <hopefully_lesser> ^status filled)
(crit_ingredient ^name <ingred_one>
                 ^value { <hopefully_greater> >= <hopefully_lesser> }
                 ^status filled)
-->
(modify <eval_criteria> ^result satisfied ^status filled)
(modify <goal> ^status complete)
(owrite (crlf)(crlf)***** Evaluation Criteria Test *****)
(owrite (crlf)<criteria_name> evaluation criteria just tested to see if
                    ingredient)
(owrite(crlf)-- <ingred_one>)
(owrite (crlf)is greater than or equal to ingredient <ingred_two> yields
              result of)
(owrite(crlf)-- satisfied)
)
```

```
(p check_dis_satisfaction_of_criteria_test
 ;*****************************************************************************
 ;  Abstract:  This is  general rule that checks for the dis-satisfaction
 ;  of the evaluation criteria.  This rule checks the opposite case of that
 ;  checked in the above rule.
 ;
 ;  Rule:
 ;  IF there is an active goal to evaluate_criteria with task <criteria_name>
 ;  AND there is an eval_criteria wme named <criteria_name> that is empty
 ;   with a supposed_greater of <greater> and supposed_lesser of <lesser>
 ;  AND there is a crit_ingredient named <greater> that is filled
 ;  AND there is a crit_ingredient named <lesser> that is filled
 ;  AND <greater> is less than <lesser>
 ;  THEN modify the eval_criteria to filled with a value in result field
 ;         of not_satisfied
 ;  AND modify the goal to complete
 ;  AND write the result
 ;*****************************************************************************
{(goal ^name evaluate_criteria ^task <criteria_name> ^status active)<goal>}
{(eval_criteria ^name <criteria_name> ^supposed_greater <ingred_one>
                                      ^supposed_lesser <ingred_two>
                                      ^result nil
                                      ^status empty)<eval_criteria>}
(crit_ingredient ^name <ingred_two> ^value <hopefully_lesser> ^status filled)
(crit_ingredient ^name <ingred_one>
                 ^value { <hopefully_greater> < <hopefully_lesser> }
                 ^status filled)
-->
(modify <eval_criteria> ^result not_satisfied ^status filled)
(modify <goal> ^status complete)
(owrite (crlf)(crlf)***** Evaluation Criteria Test *****)
(owrite (crlf)<criteria_name> evaluation criteria just tested to see if
                    ingredient)
(owrite(crlf)-- <ingred_one>)
(owrite (crlf)is greater than or equal to ingredient <ingred_two> yields
              result of)
(owrite(crlf)-- not satisfied)
)
```

A.5. Reinforced Concrete Beam Type Problem Solving Rules

The following three rules illustrate the identification of relevant parameters and criteria for evaluation of reinforced concrete beams in shear, and show computation of one of the ingredients. The first rule shows relevant parameter identification. The second rule shows relevant criteria identification. The third rule shows computation of the concrete shear strength, ΦV_c.

```
(p ident_param_for_shear
 ;*****************************************************************************
 ;  Abstract:  This rule identifies the parameters that are relevant to
 ;  solution of the shear capacity of a reinforced concrete beam.
 ;
 ;  Rule:
 ;  IF there is an active goal to identify relevant parameters
 ;  AND there are no relevant parameters yet
 ;  AND the problem definition shows a desired capacity of shear
 ;  AND the member info shows the beam type with RC_concrete material
 ;  AND the shape info has a shape filled in
 ;  THEN make the wme relevant_parameters with vector attribute param_list
 ;       to contain bw, d, fpc, Av, fy, s
 ;  AND make the list of problem relations
 ;  AND make the list of infer truths
 ;  AND make the list of infer criteria
 ;  AND make the list of acquire methods
 ;  AND make the wme rel_param_index with an index of 7
 ;  AND modify the identify goal to status complete
 ;  AND write out the relevant parameters
 ;*****************************************************************************
{(goal ^name ident_rel_param ^status active)<goal>}
-(relevant_parameters)
(problem_def ^question_of shear)
(member_info ^type beam ^material RC_concrete)
(shape_info ^shape <shape>)
-->
(make relevant_parameters ^param_list bw d f_prime_c Av fy stirrup_spacing)
(make problem_relations ^param_list i rc i rc rc rc)
(make param_infer_truths ^param_list no yes yes yes yes yes)
(make param_infer_criteria ^param_list i i year i year i)
(make param_acquire_methods ^param_list i i i compute i i)
(make rel_param_index ^index 6)
(modify <goal> ^status complete)
(owrite(crlf)(crlf)The relevant parameters for shear of this <shape> beam are)
(owrite(crlf)  bw - the width of the beam web in inches)
(owrite(crlf)  d - the member depth to steel in inches)
(owrite(crlf)  f_prime_c - the concrete strength in psi)
(owrite(crlf)  Av - the area of shear reinforcement)
(owrite(crlf)  fy - the yield strength of shear reinforcement)
(owrite(crlf)  - the stirrup spacing)
)
```

```
(p identify_general_rc_beam_shear_criteria
 ;*****************************************************************************
 ;  Abstract:  This rule fires under the identify eval criteria goal.  It
 ;  creates working memory elements with the names of the evaluation tests
 ;  to be made for checking the shear capacity of a rc_beam.
 ;
 ;  Rule:
 ;  IF there is an active goal to identify evaluation criteria
 ;  AND the member info shows member type beam with RC_concrete material
 ;  AND the problem definition shows a question of shear
 ;  THEN return the list of relevant criteria with  stirrup_spacing,
 ;     min_shear_reinforcement, no_stirrups_needed, total_shear,
 ;     raw_concrete_shear,
 ;  AND return the list of corresponding problem relations
 ;  AND return the list of supposed greater and lesser criteria ingredients
 ;  AND return the sister lists of ingredient methods of acquisition
 ;  AND make the criteria index equal to five
 ;  AND modify the identification goal to status complete
 ;  AND write that one evaluation criteria is that the strength of the raw
 ;      concrete should be checked
 ;  AND write that an evaluation criteria is that the factored applied shear
 ;      be less than the factored sum of the steel and concrete contributions
 ;  AND write that the check should be made of whether steel is even necessary
 ;  AND write that an evaluation criteria is that the minimum amount of steel
 ;      be checked
 ;  AND write that the proper spacing is required
 ;*****************************************************************************
{(goal ^name identify_eval_criteria ^status active)<goal>}
-(relevant_criteria_names)
(member_info ^type beam ^material RC_concrete)
(problem_def ^question_of shear)
(shape_info ^shape <shape> ^status filled)
-->
(make relevant_criteria_names ^criteria_list spacing min_shear_reinforcement
    no_stirrups_needed total_shear raw_concrete_shear)
(make criteria_problem_relations ^criteria_list rc rc rc rc rc)
(make supposed_greater_crit_ingredients ^criteria_list maximum_stirrup_spacing
                      Av phi_Vc_by_two phi_Vn phi_Vc)
(make supposed_greater_acquire_methods ^criteria_list compute not_compute
                                        compute compute compute)
(make supposed_lesser_crit_ingredients ^criteria_list stirrup_spacing Av_rqd
                                    Vu Vu Vu)
(make supposed_lesser_acquire_methods ^criteria_list not_compute compute
                                     not_compute not_compute not_compute)
(make rel_criteria_index ^index 5)
(modify <goal> ^status complete)
(owrite (crlf)(crlf)***The evaluation test criteria for the problem follow***)
(owrite(crlf)For the <shape> rc beam)
(owrite(crlf)(crlf)First the strength of the raw concrete is checked against
                   the applied shear)
(owrite(crlf)(crlf)Another test is that the factored sum of the concrete and
                   steel contribution)
(owrite(crlf)be greater than the factored applied shear)
(owrite(crlf)(crlf)Another test is that shear reinforcing steel is even
                  necessary)
(owrite(crlf)(crlf)Another test is that the minimum shear reinforcement be
                   met)
(owrite(crlf)(crlf)Another test is that the maximum allowable stirrup spacing
                   is not exceeded)
)

(p compute_phi_Vc
```

```
;*****************************************************************************
;  Abstract:  This rule computes the value of the shear criteria test for
;  the strength of the raw concrete.  It computes the value of the strength
;  of the raw concrete.
;
;  Rule:
;  IF there is an active goal to get ingredients with task phi_Vc
;  AND there is a crit_ingredient named phi_Vc that is empty
;  AND there is a strength reduction factor for shear
;  AND there is a value of f_prime_c
;  AND there is a value of b or bw
;  AND there is a value of d
;  THEN compute the concrete contribution by ACI 318-83 eq. 11-3
;  AND modify the ingredient to satisfied and give it the result
;*****************************************************************************
(goal ^name get_ingredients ^task raw_concrete_shear ^status active)
{(crit_ingredient ^name phi_Vc ^value nil ^status empty)<crit_ingredient>}
(strength_reduc_factor ^application shear_torsion ^phi <phi>)
(parameter_value ^name f_prime_c ^value <fpc> ^status filled)
(parameter_value ^name << b bw >> ^value <b> ^status filled)
(parameter_value ^name d ^value <d> ^status filled)
-->
(bind <root_fpc> (square-root <fpc>))
(bind <result> (compute 2 * <root_fpc> * <b> * <d> ))
(modify <crit_ingredient> ^value <result> ^status filled)
(owrite(crlf)(crlf)The capacity of just the concrete to resist shear has
                   been computed)
(owrite(crlf)-as per equation 11-3 of ACI 318-83 with the following
                   constituents)
(owrite(crlf)-- concrete yield strength of <fpc> psi)
(owrite(crlf)-- member web width of <b> inches)
(owrite(crlf)-- member depth to steel of <d> inches)
(owrite(crlf)(crlf)The shear capacity of the concrete computed to <result>
                    pounds)
)
```

A.6. Domain Declarative Knowledge

The following SEES files contain all of the permanent domain working memory knowledge.

```
;*****************************************************************************
;*****************************************************************************
;
;                    NECESSARY WORKING MEMORY ELEMENTS
;
;  Abstract:  The purpose of this file is to load in for each system run a
;  set of working memory elements that are usually of importance.  Their
;  occurance in system runs is almost definite, so they are loaded as static
;  memory items.
;
;  Source:  Joe Peters
;
;  Date:    Summer 1987
;
;*****************************************************************************
;*****************************************************************************
```

```
;=============================================================================
; The following wme's hold the information on the strength reduction factors
; used in concrete design as per section 9.3.2 of the ACI code.
(make strength_reduc_factor ^application flexure
                            ^phi 0.90        )
(make strength_reduc_factor ^application tension
                            ^phi 0.90        )
(make strength_reduc_factor ^application compression
                            ^phi 0.70        )
(make strength_reduc_factor ^application shear_torsion
                            ^phi 0.85        )
(make strength_reduc_factor ^application bearing
                            ^phi 0.70        )

;=============================================================================
; The following are the wme's for the empirical deflection criteria found on
; page 259 of Reinforced Concrete by Nawy.  The table gives the divisors for
; which the span length should be divided to give a minimum depth of beam to
; meet deflection requirements.  All results from the table should in turn be
; multiplied by (0.4 + fy/100000) where fy is in psi.
(make min_beam_thickness ^span_type simple
                         ^span_divisor 16  )
(make min_beam_thickness ^span_type fixed
                         ^span_divisor 21  )
(make min_beam_thickness ^span_type hybrid
                         ^span_divisor 18.5)
(make min_beam_thickness ^span_type cantilever
                         ^span_divisor 8   )

;=============================================================================
; The following are the history declarations for fy. According to a
; professional engineer at the NBS, only two division are required for
; the steel history.  For conservatism, only the minimum values will be used.
;   The min and max values are left here with safety factors just in case
; future prototypes wish to encorporate ranges of results. (Summer 1987)
(make steel_history ^start_year 1904 ^end_year 1970
       ^fy_min 33000 ^fy_max 50000 ^FS_min 2.06 ^FS_max 2.78)
(make steel_history ^start_year 1971 ^end_year 1986
       ^fy_min 40000 ^fy_max 60000 ^FS_min 2.0 ^FS_max 2.5)

;=============================================================================
; The following are the history declarations for f_prime_c.  According to a
; professional engineer at the NBS, three separations are about as much as
; anyone would desire for concrete.  The values used may look higher than
; what might have been poured at the time.  The feeling here is that concrete
; poured long ago is much harder now thatn it was when poured.
;    The use of minimum and maximum values is used here with factors of safety
; to leave for the possibility in later prototypes to supply ranges of
; results.  The present prototype uses only the minimum value. (Summer 1987)
(make concrete_history ^start_year 1904 ^end_year 1950
       ^f_prime_c_min 3000 ^f_prime_c_max 3000 ^FS_min 0.0 ^FS_max 0.0)
(make concrete_history ^start_year 1950 ^end_year 1970
       ^f_prime_c_min 3500 ^f_prime_c_max 3500 ^FS_min 0.0 ^FS_max 0.0)
(make concrete_history ^start_year 1970 ^end_year 1987
       ^f_prime_c_min 4000 ^f_prime_c_max 4000 ^FS_min 0.0 ^FS_max 0.0)


;=============================================================================
; The following are the history declarations for the concrete cover. (April
; 1987)
(make cover_history ^start_year 1904 ^end_year 1920
       ^use outdoor ^value 1.0)
```

```
; The next one just changes use since both uses had same cover.
(make cover_history ^start_year 1904 ^end_year 1920
       ^use indoor ^value 1.0)

;*New Time Span
(make cover_history ^start_year 1921 ^end_year 1924
       ^use outdoor ^value 2.0)
;*The next one just changes use since both uses had same cover.
(make cover_history ^start_year 1921 ^end_year 1924
       ^use indoor ^value 2.0)

;*New Time Span
(make cover_history ^start_year 1925 ^end_year 1986
       ^use indoor ^value 1.5)
(make cover_history ^start_year 1925 ^end_year 1986
       ^use outdoor ^value 2.0)

;=============================================================================
; The following is the database for reinforcing bars.
;=============================================================================

;The following are the round bar specifications.
(make rebar_specs ^shape round ^number 3 ^diameter 0.375 ^area 0.110)
(make rebar_specs ^shape round ^number 4 ^diameter 0.50 ^area 0.200)
(make rebar_specs ^shape round ^number 5 ^diameter 0.625 ^area 0.310)
(make rebar_specs ^shape round ^number 6 ^diameter 0.750 ^area 0.440)
(make rebar_specs ^shape round ^number 7 ^diameter 0.875 ^area 0.600)
(make rebar_specs ^shape round ^number 8 ^diameter 1.00 ^area 0.790)
(make rebar_specs ^shape round ^number 9 ^diameter 1.128 ^area 1.000)
(make rebar_specs ^shape round ^number 10 ^diameter 1.270 ^area 1.270)
(make rebar_specs ^shape round ^number 11 ^diameter 1.410 ^area 1.560)
(make rebar_specs ^shape round ^number 14 ^diameter 1.693 ^area 2.250)
(make rebar_specs ^shape round ^number 18 ^diameter 2.257 ^area 4.000)

; The following are the square bar specifications.
(make rebar_specs ^shape square ^number 4 ^diameter 0.5 ^area 0.25)
(make rebar_specs ^shape square ^number 8 ^diameter 1.0 ^area 1.0)
(make rebar_specs ^shape square ^number 9 ^diameter 1.125 ^area 1.265)
(make rebar_specs ^shape square ^number 10 ^diameter 1.25 ^area 1.562)
```

Appendix B
Sample Runs

This appendix supplies two example runs of the SEES expert system: one checking flexural capacity and one checking shear capacity. The flexural checking run, considers the example problem discussed in the second chapter, regarding the building in Pittsburgh being considered for alternate uses. The shear run, considers a typical T shaped beam.

B.1. Flexural Capacity

This example run shows a flexural capacity check on the beam from Figure 2-1. The problem is analyzing the 16"x26" girder for its capacity to handle two twelve thousand pound lead brick loads. The building is very old, but was originally designed for heavy loads. The input information is summarized below. A summary of the results is given after the execution log.

Problem Specification Information:

- member type = beam
- material type = reinforced concrete
- capacity to be checked = flexure at midspan
- span type = fixed
- environment = indoors
- loading = in service

Specific Problem Information:

- method of construction = detached slab
- slab thickness = approx.. 6 inches
- year built = approximately 1920
- beam width = 16 inches

- span length = 384 inches
- bay spacing of beams = 252 inches
- total height of beam = 26 inches
- load = point load of 24000 pounds

The run follows:

```
**GREETINGS**
THIS IS THE SECOND PROTOTYPE OF -SEES-

-- A KNOWLEDGE BASED EXPERT SYSTEM NAMED AFTER CHAPTER 20 OF THE ACI CODE
THAT HELPS IN THE EVALUATION OF STRENGTH OF IN SITU STRUCTURAL MEMBERS

THIS CURRENT PROTOTYPES CAPABILITY IS - REINFORCED CONCRETE BEAMS-

IN FORMULATE_PROBLEM GOAL

QUERY FOR BASIC MEMBER INFORMATION
Please submit the following information needed for the object: MEMBER_INFO
What type of member do you wish to evaluate?
   ANSWERS     CORRESPONDING VALUES

      1 ..... BEAM

      2 ..... COLUMN

Please input the number corresponding to your choice: 1

What material is the member composed of?
   ANSWERS     CORRESPONDING VALUES

      1 ..... STEEL

      2 ..... RC_CONCRETE

      3 ..... COMPOSITE

      4 ..... TIMBER

Please input the number corresponding to your choice: 2
```

```
What type of span is the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... SIMPLE

      2 ..... FIXED

      3 ..... HYBRID

      4 ..... CANTILEVER

Please input the number corresponding to your choice: 2


IN FORMULATE_PROBLEM GOAL

QUERY FOR PROBLEM DEFINITION
Please submit the following information needed for the object: PROBLEM_DEF
What capacity element are you interested in checking?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AXIAL

      2 ..... FLEXURE

      3 ..... SHEAR

      4 ..... TORSION

      5 ..... BEAM_COLUMN

Please input the number corresponding to your choice: 2

Where on the member do you need to know this?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AT_ENDS

      2 ..... MIDSPAN

Please input the number corresponding to your choice: 2


IN FORMULATE_PROBLEM GOAL

QUERY FOR IMPORTANT CRITERIA
Please submit the following information needed for the object: PROBLEM_CRITERIA
What type of loading do you wish to consider?
   ANSWERS     CORRESPONDING VALUES

      1 ..... IN_SERVICE

      2 ..... SEISMIC

Please input the number corresponding to your choice: 1

What is the exposure of the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... INDOOR
```

```
      2 ..............  OUTDOOR

Please input the number corresponding to your choice: 1


THE FORMULATE PROBLEM GOAL IS NOW COMPLETE
IN REDESIGN GOAL

THE SHAPE OF THE MEMBER MUST BE KNOWN IN ORDER TO IDENTIFY RELEVANT PARAMETERS
THE POUR INFORMATION MUST BE KNOWN TO DETERMINE THE SHAPE OF THE CONCRETE BEAM
Please submit the following information needed for the object: POUR_INFO
How is the slab connected to the beam?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  MONOLITHIC

      2 ..............  DETACHED

      3 ..............  NO_SLAB

      4 ..............  SHEAR_CONNECTORS

Please input the number corresponding to your choice: 2


MEMBER SHAPE WILL BE TREATED AS A RECTANGULAR SECTION

THE RELEVANT PARAMETERS FOR FLEXURE OF A RECTANGULAR BEAM ARE
B - THE WIDTH DIMENSION OF THE MEMBER IN INCHES
D - THE DEPTH TO TENSION STEEL OF THE MEMBER IN INCHES
D_PRIME - THE DEPTH TO COMPRESSION STEEL OF THE MEMBER IN INCHES
F_PRIME_C - THE STRENGTH OF THE CONCRETE IN PSI
FY - THE YIELD STRENGTH OF THE STEEL IN PSI
AS - THE AREA OF TENSION STEEL IN SQUARE INCHES
A_PRIME_S - THE AREA OF THE COMPRESSION STEEL IN SQUARE INCHES
BETA_ONE - THE ACI FACTOR DEFINED IN SECTION 10.2.7.3
A - THE DEPTH OF WHITNEY STRESS BLOCK
L - THE LENGTH OF THE MEMBER

DECIDE ON REINFORCING NATURE OF RC MEMBER
Please submit the following information needed for the object: DOUBLE_QUERY
Do you know that the member is doubly reinforced?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

      3 ..............  DO_NOT_KNOW

Please input the number corresponding to your choice: 3


SYSTEM ASSUMPTION OF SINGLE REINFORCEMENT FORCING
VALUE OF D_PRIME TO ZERO FOR REMAINDER OF PROBLEM

SYSTEM ASSUMPTION OF SINGLE REINFORCEMENT FORCING
VALUE OF A_PRIME_S TO ZERO FOR REMAINDER OF PROBLEM

SEE IF USER KNOWS THE VALUE OF L
Please submit the following information needed for the object: PARAMETER_QUERY
```

```
with a first attribute: NAME
having an attribute value of: L
Do you know what this design parameter equals
                                                  ?
   ANSWERS     CORRESPONDING VALUES

      1 .............. YES

      2 .............. NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: L
Please input the value of this parameter
 making sure the units comply with pounds and inches please
384.0



THE PARAMETER NAMED - A - WILL BE COMPUTED DIRECTLY
INSTEAD OF FOLLOWING NORMAL ROUTINE OF USER QUERY
**COMPUTATION WILL OCCUR AS SOON AS THE CONSTITUANT DATA ITEMS BECOME AVAILABLE

SEE IF USER KNOWS THE VALUE OF BETA_ONE
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: BETA_ONE
Do you know what this design parameter equals
                                                  ?
   ANSWERS     CORRESPONDING VALUES

      1 .............. YES

      2 .............. NO

Please input the number corresponding to your choice: 2


BETA_ONE WILL HAVE TO BE INFERRED

THE PARAMETER NAMED - AS - WILL BE COMPUTED DIRECTLY
INSTEAD OF FOLLOWING NORMAL ROUTINE OF USER QUERY
**COMPUTATION WILL OCCUR AS SOON AS THE CONSTITUANT DATA ITEMS BECOME AVAILABLE

SPECS ON THE LONGITUDINAL REINFORCING STEEL WILL NOW BE PURSUED

USER MUST BE ASKED THE REBAR NUMBER IF POSSIBLE
USER MUST ALSO BE ASKED IF SHAPE OF REBARS IS ROUND OR SQUARE

SEE IF USER KNOWS THE VALUE OF REBAR_NUMBER
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: REBAR_NUMBER
Do you know what this design parameter equals
                                                  ?
   ANSWERS     CORRESPONDING VALUES

      1 .............. YES
```

```
      2 ..............  NO

Please input the number corresponding to your choice: 2


SEE IF USER KNOWS THE VALUE OF REBAR_SHAPE
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: REBAR_SHAPE
Do you know what this design parameter equals
                                        ?
   ANSWERS      CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2


ASSUME USER DOES NOT KNOW REBAR SPECIFICATION SO
ASK THE AREA OF STEEL DIRECTLY

SEE IF USER KNOWS THE VALUE OF AS
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: AS
Do you know what this design parameter equals
                                       ?
   ANSWERS      CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2


AS WILL HAVE TO BE INFERRED

SEE IF USER KNOWS THE VALUE OF FY
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: FY
Do you know what this design parameter equals
                                       ?
   ANSWERS      CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2


FY WILL HAVE TO BE INFERRED

THE YEAR OF CONSTRUCTION MUST BE KNOWN FOR THE
```

```
ESTIMATION OF FY

SEE IF THE USER KNOWS WHEN THE STRUCTURE WAS BUILT

Please submit the following information needed for the object: YEAR_QUERY
Do you know the year give or take 5 that
the building was built?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


ASK USER WHEN STRUCTURE WAS BUILT

Please submit the following information needed for the object: YEAR_BUILT
What year was the building built? (1904 to now)
1920


FOR FY THE SYSTEM WILL USE A MINIMUM VALUE USED AROUND THE YEAR 1920
**VALUE USED WILL BE 33000 PSI
THE PARAMETER FY HAS BEEN INFERRED
TO A VALUE OF 33000

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
n


SEE IF USER KNOWS THE VALUE OF F_PRIME_C
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: F_PRIME_C
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 2


F_PRIME_C WILL HAVE TO BE INFERRED

FOR F_PRIME_C THE SYSTEM WILL ASSUME A VALUE OF 3000 PSI
FOR A BULDING BUILT AROUND 1920
THE PARAMETER F_PRIME_C HAS BEEN INFERRED
TO A VALUE OF 3000

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
y
```

```
Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: F_PRIME_C
Please input the value of this parameter
 making sure the units comply with pounds and inches please
3500



KNOWING VALUE OF F_PRIME_C
BETA_ONE GETS A VALUE OF 0.85 AS PER ACI 318-83
THE PARAMETER BETA_ONE HAS BEEN INFERRED
TO A VALUE OF 0.85

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
n


SEE IF USER KNOWS THE VALUE OF D
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: D
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2



D WILL HAVE TO BE INFERRED

SEE IF USER KNOWS THE VALUE OF H
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: H
Do you know what this design parameter equals
                                   ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: H
Please input the value of this parameter
 making sure the units comply with pounds and inches please
26



SEE IF USER KNOWS THE VALUE OF COVER
```

```
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: COVER
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: COVER
Please input the value of this parameter
 making sure the units comply with pounds and inches please
1.5



USER DOES NOT KNOW THE REBAR DIAMETER
VALUE WILL BE DEFAULTED TO THAT OF A NUM 8 BAR


THE DESIGN PARAMETER D THE DEPTH TO STEEL CAN
NOW BE COMPUTED WITH KNOWLEDGE OF THE FOLLOWING
COVER EQUALS 1.5 INCHES
BAR DIAMETER IS OR ASSUMED TO BE 1.0 INCHES
MEMBER HEIGHT IS 26 INCHES
D EQUALS H MINUS THE COVER MINUS BAR DIAMETER OVER TWO

D EQUALS 24.0 INCHES

SEE IF USER KNOWS THE VALUE OF B
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: B
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: B
Please input the value of this parameter
 making sure the units comply with pounds and inches please
16



USER UNABLE TO SPECIFY AREA OF STEEL INFORMATION
**SYSTEM WILL ASSUME EXISTENCE OF AT LEAST ONE PERCENT STEEL IN MEMBER
```

```
AREA OF TENSION STEEL USED COMPUTED FROM WEB WIDTH OF 16 INCHES AND MEMBER
DEPTH OF 24.0 INCHES
**ONE PERCENT YIELDS A STEEL AREA OF 3.84 SQUARE INCHES
THE PARAMETER AS HAS BEEN INFERRED
TO A VALUE OF 3.84

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
n


DEPTH OF WHITNEY STRESS BLOCK COMPUTED USING
A_SUB_S = 3.84 SQUARE INCHES AND A_PRIME_SF = 0 SQUARE INCHES
FY = 33000 PSI AND F_PRIME_C = 3500 PSI
AND THE RECTANGULAR MEMBER WIDTH B 16 INCHES
**YIELDS 2.66218 INCHES FOR THE DEPTH OF THE STRESS BLOCK

***THE EVALUATION TEST CRITERIA FOR THE PROBLEM FOLLOW***
ONE EVALUATION CRITERIA IS THAT THE FACTORED NOMINAL
MOMENT CAPACITY PHI_MN MUST BE GREATER THAN THE FACTORED APPLIED MOMENT MU

ANOTHER EVALUATION CRITERIA IS THAT RHO BE LESS THAN 75 PERCENT OF RHO BALANCED

ANOTHER EVALUATION CRITERIA IS THAT RHO BE GREATER THAN RHO MINIMUM

ANOTHER EVALUATION CRITERIA IS THAT THE MEMBER DEPTH BE GREATER THAN
THE MINIMUM DEPTH COMPUTED FROM EMPIRICAL DEFLECTION CRITERIA

VALUE OF MINIMUM REINFORCEMENT RATIO RHO_MIN COMPUTED AS PER ACI 318-83
EQUATION 10-3
EQUATES TO 6.06061F-03

THE VALUE OF RHO HAS BEEN COMPUTED FOR THE TENSION STEEL
USING AN AREA OF TENSION STEEL OF 3.84 SQUARE INCHES
AND A MEMBER WEB WIDTH OF 16 INCHES AND MEMBER DEPTH TO STEEL OF 24.0 INCHES
**VALUE OF RHO EQUATES TO 0.01

***** EVALUATION CRITERIA TEST *****
MIN_STEEL EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- RHO
IS GREATER THAN OR EQUAL TO INGREDIENT RHO_MIN YIELDS RESULT OF
-- SATISFIED

THE BALANCED REINFORCEMENT RATIO RHO_B COMPUTED AS PER
ACI 318-83 EQUATION 8-1
USING A BETA_ONE VALUE OF 0.85
FY OF 33000 PSI AND F_PRIME_C OF 3500 PSI
EVALUATES TO 5.55559F-02

***** EVALUATION CRITERIA TEST *****
DUCTILE_FAILURE EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- 75%RHO_B
IS GREATER THAN OR EQUAL TO INGREDIENT RHO YIELDS RESULT OF
-- SATISFIED

CRITERIA INGREDIENT MU TO BE OBTAINED THRU GET_PARAMETER METHODS

SEE IF USER KNOWS THE VALUE OF MU
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: MU
Do you know what this design parameter equals
```

```
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..... YES

      2 ..... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: MU
Please input the value of this parameter
 making sure the units comply with pounds and inches please
3120000



CRITERIA INGREDIENT MU HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED PARAMETER
MU WITH A VALUE OF 3120000

THE MOMENT CAPACITY IS BASED ON DUCTILE FAILURE
IT HAS BEEN COMPUTED WITH
STRENGTH REDUCTION FACTOR OF 0.9
AREA OF TENSION STEEL OF 3.84 SQUARE INCHES
AREA OF COMPRESSION STEEL OF 0 SQUARE INCHES
STEEL YIELD STRENGTH OF 33000 PSI
DEPTH OF STRESS BLOCK OF 2.66218 INCHES
DEPTH TO TENSION STEEL OF 24.0 INCHES
DEPTH TO COMPRESSION STEEL OF 0 INCHES

THE MOMENT CAPACITY EQUATES TO 2.58534F+06 POUND-INCHES

***** EVALUATION CRITERIA TEST *****
MOMENT EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- PHI_MN
IS GREATER THAN OR EQUAL TO INGREDIENT MU YIELDS RESULT OF
-- NOT SATISFIED

THE MINIMUM EXPECTED BEAM DEPTH FOR MEETING EMPIRICAL
DEFLECTION CRITERIA COMPUTED AS PER TABLE 9-5A OF ACI 318-83
FOR A SPAN LENGTH OF 384.0 INCHES
AND SPAN TYPE OF FIXED
AND AN FY OF 33000 PSI
EVALUATES TO 13.3486 INCHES

CRITERIA INGREDIENT D HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED PARAMETER
D WITH A VALUE OF 24.0

***** EVALUATION CRITERIA TEST *****
DEFLECTION EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- D
IS GREATER THAN OR EQUAL TO INGREDIENT MIN_DEPTH YIELDS RESULT OF
-- SATISFIED

********** ECHO PROBLEM DEFINITION **********
THE FIXED RC_CONCRETE RECTANGLE BEAM UNDER
IN_SERVICE LOADING CONDITIONS IN AN INDOOR ENVIRONMENT
HAS BEEN INVESTIGATED FOR A QUESTION OF FLEXURE AT MIDSPAN
```

```
IN SOLUTION TO THE SPECIFIED PROBLEM THE FOLLOWING RESULTS WERE
OBTAINED REGARDING THE CRITERIA RELEVANT TO THE DESIRED CAPACITY
*SYSTEM UNITS ARE POUNDS AND INCHES

THE TEST CRITERIA NAMED - DEFLECTION - WHERE IT IS DESIRED
THAT THE VALUE OF D BE GREATER THAN THE VALUE OF MIN_DEPTH

HAS YIELDED THE FOLLOWING
- D HAS A VALUE OF 24.0
- MIN_DEPTH HAS A VALUE OF 13.3486
THE RESULT OF THIS TEST BEING -- SATISFIED

THE TEST CRITERIA NAMED - MOMENT - WHERE IT IS DESIRED
THAT THE VALUE OF PHI_MN BE GREATER THAN THE VALUE OF MU

HAS YIELDED THE FOLLOWING
- PHI_MN HAS A VALUE OF 2.58534F+06
- MU HAS A VALUE OF 3120000
THE RESULT OF THIS TEST BEING -- NOT_SATISFIED

THE TEST CRITERIA NAMED - DUCTILE_FAILURE - WHERE IT IS DESIRED
THAT THE VALUE OF 75%RHO_B BE GREATER THAN THE VALUE OF RHO

HAS YIELDED THE FOLLOWING
- 75%RHO_B HAS A VALUE OF 5.55559F-02
- RHO HAS A VALUE OF 0.01
THE RESULT OF THIS TEST BEING -- SATISFIED

THE TEST CRITERIA NAMED - MIN_STEEL - WHERE IT IS DESIRED
THAT THE VALUE OF RHO BE GREATER THAN THE VALUE OF RHO_MIN

HAS YIELDED THE FOLLOWING
- RHO HAS A VALUE OF 0.01
- RHO_MIN HAS A VALUE OF 6.06061F-03
THE RESULT OF THIS TEST BEING -- SATISFIED


-- SEES -- WILL NOW TERMINATE

THIS CURRENT VERSION OF SEES IS THE SECOND PROTOTYPE
AND APOLOGIZES FOR THE LACK OF ANTICAPATED CAPABILITIES
OF THE FULL INTENDED SYSTEM
end -- no production true
 104 PRODUCTIONS (1263 // 2201 NODES)
 109 FIRI
```

Summary of results:

- The minimum depth of member requirement was satisfied; the actual member depth 24 inches was greater than the empirical required 13 inches.
- The reinforcement requirements were satisfied. This is due to the fact that the system had to infer a value of 1% for the reinforcement ratio.
- The moment capacity check was not satisfied. The member moment capacity was 2,590,000 inch-pounds and the factored applied moment was approximately 3,120,000

inch-pounds. The applied moment was computed before hand as the lead brick moment at midspan plus the dead load moment at midspan, increased by 15%.

- Perhaps the execution should be run again assuming the slab is attached to the beam.

B.2. Shear Capacity

This example run shows a shear capacity check on a T shaped beam for which the user knows more about the beam than just member dimensions. The input is summarized below. A summary of system results follows the run.

Problem Specification Information:

- member type = beam
- material type = reinforced concrete
- capacity to be checked = shear at ends
- span type = simple
- environment = outdoor
- loading = in service

Specific Problem Information:

- method of construction = shear connectors
- location of member in slab = edge
- year built = approximately 1970
- steel yield strength = 40,000 psi
- stirrups = #4 round bar at 5 inches
- beam web width = 16 inches
- span length = 360 inches
- total height of beam = 24 inches
- load = point load of 64500 pounds
- location of load = at ends

The system run follows.

```
**GREETINGS**
THIS IS THE SECOND PROTOTYPE OF -SEES-

-- A KNOWLEDGE BASED EXPERT SYSTEM NAMED AFTER CHAPTER 20 OF THE ACI CODE
THAT HELPS IN THE EVALUATION OF STRENGTH OF IN SITU STRUCTURAL MEMBERS

THIS CURRENT PROTOTYPES CAPABILITY IS - REINFORCED CONCRETE BEAMS-

IN FORMULATE_PROBLEM GOAL

QUERY FOR BASIC MEMBER INFORMATION
Please submit the following information needed for the object: MEMBER_INFO
What type of member do you wish to evaluate?
   ANSWERS     CORRESPONDING VALUES

      1 ..... BEAM

      2 ..... COLUMN

Please input the number corresponding to your choice: 1

What material is the member composed of?
   ANSWERS     CORRESPONDING VALUES

      1 ..... STEEL

      2 ..... RC_CONCRETE

      3 ..... COMPOSITE

      4 ..... TIMBER

Please input the number corresponding to your choice: 2

What type of span is the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... SIMPLE

      2 ..... FIXED

      3 ..... HYBRID

      4 ..... CANTILEVER

Please input the number corresponding to your choice: 1
```

```
IN FORMULATE_PROBLEM GOAL

QUERY FOR PROBLEM DEFINITION
Please submit the following information needed for the object: PROBLEM_DEF
What capacity element are you interested in checking?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AXIAL

      2 ..... FLEXURE

      3 ..... SHEAR

      4 ..... TORSION

      5 ..... BEAM_COLUMN

Please input the number corresponding to your choice: 3

Where on the member do you need to know this?
   ANSWERS     CORRESPONDING VALUES

      1 ..... AT_ENDS

      2 ..... MIDSPAN

Please input the number corresponding to your choice: 1


IN FORMULATE_PROBLEM GOAL

QUERY FOR IMPORTANT CRITERIA
Please submit the following information needed for the object: PROBLEM_CRITERIA
What type of loading do you wish to consider?
   ANSWERS     CORRESPONDING VALUES

      1 ..... IN_SERVICE

      2 ..... SEISMIC

Please input the number corresponding to your choice: 1

What is the exposure of the member?
   ANSWERS     CORRESPONDING VALUES

      1 ..... INDOOR

      2 ..... OUTDOOR

Please input the number corresponding to your choice: 2


THE FORMULATE PROBLEM GOAL IS NOW COMPLETE
IN REDESIGN GOAL

THE SHAPE OF THE MEMBER MUST BE KNOWN IN ORDER TO IDENTIFY RELEVANT PARAMETERS
THE POUR INFORMATION MUST BE KNOWN TO DETERMINE THE SHAPE OF THE CONCRETE BEAM
Please submit the following information needed for the object: POUR_INFO
How is the slab connected to the beam?
   ANSWERS     CORRESPONDING VALUES
```

```
      1 ..............  MONOLITHIC

      2 ..............  DETACHED

      3 ..............  NO_SLAB

      4 ..............  SHEAR_CONNECTORS

Please input the number corresponding to your choice: 4


LOCATION OF MEMBER UNDER SLAB IS IMPORTANT
Please submit the following information needed for the object: LOCATION_INFO
What is the location of the member within the structure?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  INTERIOR

      2 ..............  EDGE

Please input the number corresponding to your choice: 2


MEMBER SHAPE WILL BE TREATED AS A SPANDREL

THE RELEVANT PARAMETERS FOR SHEAR OF THIS SPANDREL BEAM ARE
BW - THE WIDTH OF THE BEAM WEB IN INCHES
D - THE MEMBER DEPTH TO STEEL IN INCHES
F_PRIME_C - THE CONCRETE STRENGTH IN PSI
AV - THE AREA OF SHEAR REINFORCEMENT
FY - THE YIELD STRENGTH OF SHEAR REINFORCMENT
- THE STIRRUP SPACING

SEE IF USER KNOWS THE VALUE OF STIRRUP_SPACING
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: STIRRUP_SPACING
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: STIRRUP_SPACING
Please input the value of this parameter
 making sure the units comply with pounds and inches please
5



SEE IF USER KNOWS THE VALUE OF FY
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: FY
Do you know what this design parameter equals
```

```
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: FY
Please input the value of this parameter
 making sure the units comply with pounds and inches please
40000



THE PARAMETER NAMED - AV - WILL BE COMPUTED DIRECTLY
INSTEAD OF FOLLOWING NORMAL ROUTINE OF USER QUERY
**COMPUTATION WILL OCCUR AS SOON AS THE CONSTITUANT DATA ITEMS BECOME AVAILABLE

SPECS ON THE SHEAR REINFORCING STEEL WILL NOW BE PURSUED

USER MUST BE ASKED THE STIRRUP REBAR NUMBER IF POSSIBLE
USER MUST ALSO BE ASKED IF SHAPE OF REBARS IS ROUND OR SQUARE

SEE IF USER KNOWS THE VALUE OF STIRRUP_REBAR_NUMBER
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: STIRRUP_REBAR_NUMBER
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: STIRRUP_REBAR_NUMBER
Please input the value of this parameter
 making sure the units comply with pounds and inches please
4



SEE IF USER KNOWS THE VALUE OF STIRRUP_REBAR_SHAPE
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: STIRRUP_REBAR_SHAPE
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES
```

```
      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: STIRRUP_REBAR_SHA
PE_INFO
What is the shape of the stirrup reinforcing steel?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  ROUND

      2 ..............  SQUARE

Please input the number corresponding to your choice: 1



FOR THE ROUND NUMBER 4 BAR A DIAMETER OF 0.5
INCHES AND BAR AREA OF 0.2 SQUARE INCHES IS USED

THE AREA OF SHEAR REINFORCING STEEL A_SUB_V AS DEFINED IN ACI 318-83
KNOWING A SHEAR REBAR SPEC OF NUMBER 4 BAR WITH 0.2 SQUARE INCH CROSS
SECTION YIELDS AN A_SUB_V OF 0.4 SQUARE INCHES

SEE IF USER KNOWS THE VALUE OF F_PRIME_C
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: F_PRIME_C
Do you know what this design parameter equals
                                    ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2



F_PRIME_C WILL HAVE TO BE INFERRED

THE YEAR OF CONSTRUCTION MUST BE KNOWN FOR THE
ESTIMATION OF F_PRIME_C

SEE IF THE USER KNOWS WHEN THE STRUCTURE WAS BUILT

Please submit the following information needed for the object: YEAR_QUERY
Do you know the year give or take 5 that
the building was built?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1



ASK USER WHEN STRUCTURE WAS BUILT
```

```
Please submit the following information needed for the object: YEAR_BUILT
What year was the building built? (1904 to now)
1970


FOR F_PRIME_C THE SYSTEM WILL ASSUME A VALUE OF 4000 PSI
FOR A BULDING BUILT AROUND 1970
THE PARAMETER F_PRIME_C HAS BEEN INFERRED
TO A VALUE OF 4000

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
n


SEE IF USER KNOWS THE VALUE OF D
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: D
Do you know what this design parameter equals
                                              ?
   ANSWERS      CORRESPONDING VALUES

      1 ............... YES

      2 ............... NO

Please input the number corresponding to your choice: 2


D WILL HAVE TO BE INFERRED

SEE IF USER KNOWS THE VALUE OF H
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: H
Do you know what this design parameter equals
                                              ?
   ANSWERS      CORRESPONDING VALUES

      1 ............... YES

      2 ............... NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: H
Please input the value of this parameter
 making sure the units comply with pounds and inches please
24.0


SEE IF USER KNOWS THE VALUE OF COVER
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: COVER
```

```
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2



COVER WILL HAVE TO BE INFERRED

THE VALUE OF CONCRETE COVER TYPICALLY USED IN THE YEAR
1970 FOR OUTDOOR USE WAS 2.0 INCHES
THIS VALUE WILL BE USED FOR COVER IN THIS PROBLEM

THE PARAMETER COVER HAS BEEN INFERRED
TO A VALUE OF 2.0

DO YOU WISH TO OVER-RIDE THIS VALUE WITH ONE OF YOUR OWN?
Y OR N
Y

Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: COVER
Please input the value of this parameter
 making sure the units comply with pounds and inches please
1.5



SPECS ON THE LONGITUDINAL REINFORCING STEEL WILL NOW BE PURSUED

USER MUST BE ASKED THE REBAR NUMBER IF POSSIBLE
USER MUST ALSO BE ASKED IF SHAPE OF REBARS IS ROUND OR SQUARE

SEE IF USER KNOWS THE VALUE OF REBAR_NUMBER
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: REBAR_NUMBER
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2



USER DOES NOT KNOW THE REBAR DIAMETER
VALUE WILL BE DEFAULTED TO THAT OF A NUM 8 BAR


SEE IF USER KNOWS THE VALUE OF REBAR_SHAPE
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
```

```
having an attribute value of: REBAR_SHAPE
Do you know what this design parameter equals
                                        ?
   ANSWERS      CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 2


THE DESIGN PARAMETER D THE DEPTH TO STEEL CAN
NOW BE COMPUTED WITH KNOWLEDGE OF THE FOLLOWING
COVER EQUALS 1.5 INCHES
BAR DIAMETER IS OR ASSUMED TO BE 1.0 INCHES
MEMBER HEIGHT IS 24.0 INCHES
D EQUALS H MINUS THE COVER MINUS BAR DIAMETER OVER TWO

D EQUALS 22.0 INCHES

SEE IF USER KNOWS THE VALUE OF BW
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: BW
Do you know what this design parameter equals
                                        ?
   ANSWERS      CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: BW
Please input the value of this parameter
 making sure the units comply with pounds and inches please
16.0


***THE EVALUATION TEST CRITERIA FOR THE PROBLEM FOLLOW***
FOR THE SPANDREL RC BEAM

FIRST THE STRENGTH OF THE RAW CONCRETE IS CHECKED AGAINST THE APPLIED SHEAR

ANOTHER TEST IS THAT THE FACTORED SUM OF THE CONCRETE AND STEEL CONTRIBUTION
BE GREATER THAN THE FACTORED APPLIED SHEAR

ANOTHER TEST IS THAT SHEAR REINFORCING STEEL IS EVEN NECESSARY

ANOTHER TEST IS THAT THE MINIMUM SHEAR REINFORCEMENT BE MET

ANOTHER TEST IS THAT THE MAXIMUM ALLOWABLE STIRRUP SPACING IS NOT EXCEEDED

CRITERIA INGREDIENT VU TO BE OBTAINED THRU GET_PARAMETER METHODS
```

```
SEE IF USER KNOWS THE VALUE OF VU
Please submit the following information needed for the object: PARAMETER_QUERY
with a first attribute: NAME
having an attribute value of: VU
Do you know what this design parameter equals
                                        ?
   ANSWERS     CORRESPONDING VALUES

      1 ..............  YES

      2 ..............  NO

Please input the number corresponding to your choice: 1


Please submit the following information needed for the object: PARAMETER_VALUE
with a first attribute: NAME
having an attribute value of: VU
Please input the value of this parameter
 making sure the units comply with pounds and inches please
64500



CRITERIA INGREDIENT VU HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED PARAMETER
VU WITH A VALUE OF 64500

THE CAPACITY OF JUST THE CONCRETE TO RESIST SHEAR HAS BEEN COMPUTED
-AS PER EQUATION 11-3 OF ACI 318-83 WITH THE FOLLOWING CONSTITUANTS
-- CONCRETE YIELD STRENGTH OF 4000 PSI
-- MEMBER WEB WIDTH OF 16.0 INCHES
-- MEMBER DEPTH TO STEEL OF 22.0 INCHES

THE SHEAR CAPACITY OF THE CONCRETE COMPUTED TO 44524.9 POUNDS

***** EVALUATION CRITERIA TEST *****
RAW_CONCRETE_SHEAR EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- PHI_VC
IS GREATER THAN OR EQUAL TO INGREDIENT VU YIELDS RESULT OF
-- NOT SATISFIED

THE NOMINAL SHEAR CAPACITY OF THE MEMBER HAS BEEN COMPUTED
BY ACI EQUATION 11-2 AND MULTIPLIED BY A STRENGTH REDUCTION FACTOR
**THE PARAMETERS USED WERE
- A PHI FACTOR OF 0.85
- CONCRETE YIELD STRENGTH OF 4000 PSI AND - STEEL YIELD OF 40000 PSI
- MEMBER WEB WIDTH OF 16.0 INCHES AND - MEMBER DEPTH TO STEEL OF 22.0 INCHES
- SHEAR STEEL AREA OF 0.4 SQUARE INCHES
- STIRRUP SPACING OF 5 INCHES

THE CONCRETE CONTRIBUTION WAS COMPUTED FROM ACI EQUATION 11-3
-- AND COMPUTED TO 44524.9 POUNDS
THE STEEL CONTRIBUTION WAS COMPUTED AS THE MINIMUM OF EITHER
ACI EQUATION 11-17 OR PARAGRAPH 11-5-6-8
-- AND COMPUTES TO 70400.0 POUNDS

THE TOTAL FACTORED NOMINAL SHEAR CAPACITY COMES TO 97686.1 POUNDS

***** EVALUATION CRITERIA TEST *****
TOTAL_SHEAR EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- PHI_VN
IS GREATER THAN OR EQUAL TO INGREDIENT VU YIELDS RESULT OF
```

```
-- SATISFIED

EVALUATION TEST CRITERIA INGREDIENT - PHI_VC_BY_TWO - HAS BEEN CALCULATED

HALF OF THE FACTORED NOMINAL SHEAR CAPACITY IS 22262.4 POUNDS

***** EVALUATION CRITERIA TEST *****
NO_STIRRUPS_NEEDED EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- PHI_VC_BY_TWO
IS GREATER THAN OR EQUAL TO INGREDIENT VU YIELDS RESULT OF
-- NOT SATISFIED

THE REQUIRED SHEAR REINFORCING STEEL AREA AS PER ACI EQUATION 11-14 USING
- A MEMBER WEB WIDTH OF 16.0 INCHES
- A STIRRUP SPACING OF 5 INCHES
- A STEEL YIELD STRENGTH OF 40000 PSI

YIELDS A RESULT OF 0.1 SQUARE INCHES

CRITERIA INGREDIENT AV HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED PARAMETER
AV WITH A VALUE OF 0.4

***** EVALUATION CRITERIA TEST *****
MIN_SHEAR_REINFORCEMENT EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- AV
IS GREATER THAN OR EQUAL TO INGREDIENT AV_RQD YIELDS RESULT OF
-- SATISFIED

CRITERIA INGREDIENT STIRRUP_SPACING HAS BEEN GIVEN VALUE OF PREVIOUSLY OBTAINED
PARAMETER
STIRRUP_SPACING WITH A VALUE OF 5

THE MAXIMUM ALLOWABLE STIRRUP SPACING AS PER ACI 318-83
KNOWING THAT D EQUALS 22.0 INCHES EQUATES TO 11.0 INCHES

***** EVALUATION CRITERIA TEST *****
SPACING EVALUATION CRITERIA JUST TESTED TO SEE IF INGREDIENT
-- MAXIMUM_STIRRUP_SPACING
IS GREATER THAN OR EQUAL TO INGREDIENT STIRRUP_SPACING YIELDS RESULT OF
-- SATISFIED

********** ECHO PROBLEM DEFINITION **********
THE SIMPLE RC_CONCRETE SPANDREL BEAM UNDER
IN_SERVICE LOADING CONDITIONS IN AN OUTDOOR ENVIRONMENT
HAS BEEN INVESTIGATED FOR A QUESTION OF SHEAR AT AT_ENDS


IN SOLUTION TO THE SPECIFIED PROBLEM THE FOLLOWING RESULTS WERE
OBTAINED REGARDING THE CRITERIA RELEVANT TO THE DESIRED CAPACITY
*SYSTEM UNITS ARE POUNDS AND INCHES

THE TEST CRITERIA NAMED - SPACING - WHERE IT IS DESIRED
THAT THE VALUE OF MAXIMUM_STIRRUP_SPACING BE GREATER THAN THE VALUE OF STIRRUP_S
PACING

HAS YIELDED THE FOLLOWING
- MAXIMUM_STIRRUP_SPACING HAS A VALUE OF 11.0
- STIRRUP_SPACING HAS A VALUE OF 5
THE RESULT OF THIS TEST BEING -- SATISFIED

THE TEST CRITERIA NAMED - MIN_SHEAR_REINFORCEMENT - WHERE IT IS DESIRED
THAT THE VALUE OF AV BE GREATER THAN THE VALUE OF AV_RQD
```

```
HAS YIELDED THE FOLLOWING
- AV HAS A VALUE OF 0.4
- AV_RQD HAS A VALUE OF 0.1
THE RESULT OF THIS TEST BEING -- SATISFIED

THE TEST CRITERIA NAMED - NO_STIRRUPS_NEEDED - WHERE IT IS DESIRED
THAT THE VALUE OF PHI_VC_BY_TWO BE GREATER THAN THE VALUE OF VU

HAS YIELDED THE FOLLOWING
- PHI_VC_BY_TWO HAS A VALUE OF 22262.4
- VU HAS A VALUE OF 64500
THE RESULT OF THIS TEST BEING -- NOT_SATISFIED

THE TEST CRITERIA NAMED - TOTAL_SHEAR - WHERE IT IS DESIRED
THAT THE VALUE OF PHI_VN BE GREATER THAN THE VALUE OF VU

HAS YIELDED THE FOLLOWING
- PHI_VN HAS A VALUE OF 97686.1
- VU HAS A VALUE OF 64500
THE RESULT OF THIS TEST BEING -- SATISFIED

THE TEST CRITERIA NAMED - RAW_CONCRETE_SHEAR - WHERE IT IS DESIRED
THAT THE VALUE OF PHI_VC BE GREATER THAN THE VALUE OF VU

HAS YIELDED THE FOLLOWING
- PHI_VC HAS A VALUE OF 44524.9
- VU HAS A VALUE OF 64500
THE RESULT OF THIS TEST BEING -- NOT_SATISFIED


-- SEES -- WILL NOW TERMINATE

THIS CURRENT VERSION OF SEES IS THE SECOND PROTOTYPE
AND APOLOGIZES FOR THE LACK OF ANTICAPATED CAPABILITIES
OF THE FULL INTENDED SYSTEM
end -- no production true
 104 PRODUCTIONS (1263 // 2201 NODES)
 107 FIRINGS (380 RHS ACTIONS)
 0 MEAN WORKING MEMORY SIZE (174 MAXIMUM)
 0 MEAN CONFLICT SET SIZE (11 MAXIMUM)
 0 MEAN TOKEN MEMORY SIZE (304 MAXIMUM)

NIL
*
Going to DOS, type EXIT to return to GCLISP.

The IBM Personal Computer DOS
Version 3.10 (C)Copyright International Business Machines Corp 1981, 1985
             (C)Copyright Microsoft Corp 1981, 1985
```

Summary of results:

- The test of the raw_concrete_shear strength was not satisfied with an allowable shear capacity of 45,000 pounds and an applied load of 65,000 pounds.
- The no_stirrups_needed test is not satisfied as well since the applied load is greater than half of the concrete allowable.

- The total_shear test was satisfied. When contribution of the shear reinforcing steel was considered, the allowable shear capacity was increased to 98,000 pounds.
- The check for minimum shear reinforcement was satisfied: the actual shear steel are being 0.4 inches and the required minimum being 0.1 inches.
- The maximum_stirrup_spacing test was satisfied: the maximum allowable stirrup spacing equaling 11 inches and the actual stirrup spacing equaling 5 inches.

References

[Abbott 86] Abbott, Russell J.
Software Development.
Wiley Interscience, New York, New York, 1986.

[ACI 19 10] National Association of Cement Users.
Standard Building Regulations for the Use of Reinforced Concrete Standard No. 4.
In *Proceedings of the American Concrete Institute Volume 6*, pages 349-361. American Concrete Institute, Philidelphia, Pennsylvania, February, 1910.

[ACI 19 20] American Concrete Institute.
Standard Building Regulations for the Use of Reinforced Concrete Standard No. 23.
In *Proceedings of the American Concrete Institute Volume 16*, pages 283-302. American Concrete Institute, Detroit, Michigan, April, 1920.

[ACI 19 28] American Concrete Institute.
Proposed Standard Building Regulations for Reinforced Concrete.
American Concrete Institute, Detroit, Michigan, 1928.

[ACI 19 36] American Concrete Institute.
Building Regulations for Reinforced Concrete A.C.I. 501-36-T.
American Concrete Institute, Detroit, Michigan, 1936.

[ACI 19 41] American Concrete Institute.
Building Regulations for Reinforced Concrete ACI 318-41.
American Concrete Institute, Detroit, Michigan, 1941.

[ACI 19 48] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-47.
American Concrete Institute, Detroit, Michigan, 1948.

[ACI 19 51] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-51.
American Concrete Institute, Detroit, Michigan, 1951.

[ACI 19 56] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-56.
American Concrete Institute, Detroit, Michigan, 1956.

[ACI 19 63] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-63.
American Concrete Institute, Detroit, Michigan, 1963.

[ACI 19 77] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-77.
American Concrete Institute, Detroit, Michigan, 1977.

[ACI 19 83] American Concrete Institute.
Building Code Requirements for Reinforced Concrete ACI 318-83.
American Concrete Institute, Detroit, Michigan, 1983.

[Brownston et al 86]
Brownston, Lee and Robert Farrell, Elaine Kant, Nancy Martin.
Programming Expert Systems in OPS5.
Addison-Wesley, Reading, Massachusetts, 1986.

[Buchanan et al 83]
Buchanan, Bruce G. et al.
Constructing an Expert System.
In *Building Expert Systems*, chapter 5, pages 127-167. Addison-Wesley, 1983.

[Fenves et al 87] Fenves, Steven J. and Nelson Baker, James Garrett, Duv Sriram, George Turkiyyah.
Expert Systems in Civil Engineering Course Notes.
Department of Civil Engineering Carnegie Mellon University, Pittsburgh, Pennsylvania, 1987.

[Forgy 81] Forgy, Charles L.
OPS5 User's Manual
Pittsburgh, Pennsylvania, 1981.

[Furguson 79] Furguson, Phil M.
Reinforced Concrete Fundamentals.
John Wiley & Sons, New York, New York, 1979.

[Gaylord,Gaylord 79]
Gaylord, Edwin H., Jr. and Charles N. Gaylord (editor).
Structural Engineering Handbook 2cnd Edition.
McGraw Hill, New York, New York, 1979.

[Harmon,King 85] Harmon, Paul and David King.
Expert Systems.
Wiley Press, New York, New York, 1985.

[Hool 17] Hool, George A.
Reinforced Concrete Construction Volume 1.
McGraw Hill, New York, New York, 1917.

[Kerekes,Reid 54] Kerekes, Frank and Harold B. Reid.
Fifty Years of Development in Building Code Requirements for Reinforced Concrete.
Journal of the American Concrete Institute 25(6):441-471, February, 1954.

[Kostem,Maher 86]
Kostem, Celal N. and Mary Lou Maher (editor).
Expert Systems in Civil Engineering.
American Society of Civil Engineers, New York, New York, 1986.

[Lopez,Elam 87] Lopez, L.A. and S. Elam.
SICAD, Standards Interface for Computer Aided Design.
Technical Report Government Contract Report 87-531, National Bureau of Standards, Gaithersburg, Maryland, May, 1987.

[Maher 86] Maher, Mary Lou.
Problem Solving Using Expert System Techniques.
In *Expert Systems in Civil Engineering*, chapter 2, pages 7-17. American Society of Civil Engineers, New York, New York, 1986.

[Nawy 85] Nawy, Edward G.
Reinforced Concrete.
Prentice Hall, Englewood Cliffs, New Jersey, 1985.

[Reynolds 39] Reynolds, Chas E.
Reinforced Concrete Designers Handbook 2cnd Edition.
Concrete Publications Limited, London, 1939.

[Steele 84] Steele, Guy L.
Common LISP.
Digital Press, Bedford, Massachusetts, 1984.

[Urquhart,O'Rourke 35]
Urquhart, Leonard Church and Charles Edward O'Rourke.
Design of Concrete Structures.
McGraw Hill, New York, New York, 1935.

[Wang,Salmon 79] Wang, Chu-Kia and Charles G. Salmon.
Reinforced Concrete Design, 3rd Edition.
Harper & Row, 1979.

[Winter 82] Winter, George.
Development of a National Building Code for Reinforced Concrete 1908-1977.
Concrete International :27-37, December, 1982.

[Winter & Nilson 79]
Winter, George and Arthur H. Nilson.
Design of Concrete Structures.
McGraw Hill, New York, New York, 1979.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS-GCR-87-538	2. Performing Organ. Report No.	3. Publication Date JANUARY 1988

4. TITLE AND SUBTITLE

SEES: An Expert System for the Strength Evaluation of Existing Structural Members

5. AUTHOR(S)

Joseph Francis Peters

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.
70NANB5H0584

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This thesis is a report of the design and implementation of the SEES expert system; a knowledge based system for strength evaluation of existing structural members. The expert system provides engineering knowledge for in situ member evaluation compiled into an interactive computer program to aid in the solution of a characteristically uncertain engineering problem. Not intended to replace engineers the system's purpose is twofold: provide engineering judgment in a computer program, and attend to details that engineers may let go unnoticed.

Two studies made up the work of this thesis: the use of an artificial intelligence programming environment for an engineering application and the study of expert domain knowledge needed for strength evaluation. The resulting computer program is a goal driven expert system developed in a PC-AT version of OPS5, where the problem solution involves the satisfaction of three top level goals: formulation, redesign, and evaluation. Formulation provides the problem definition. Redesign requests or infers design parmeters. Evaluation uses current codes to assess the capacity of the existing member. The study of expert domain knowledge involved learning the procedures and rules of engineering judgment used in the evaluation of flexural, shear, and torsional capacity of reinforced concrete T, spandrel, and rectangular beam sections.

The result of the study is a generic engineering expert system control structure readily expandable to at least seven other strength evaluation problems and a prototype implementation for reinforced concrete beams.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

expert system, knowledge based system, strength evaluation, structural members

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
136

15. Price
$18.95